MASTER EN SCIENCES INFORMATIQUES

Logiciel de simulation d'un jeu de marionnettes pour enfants infirmes moteurs cérébraux

Brousmiche, Laurence; Gillet, Xavier

Award date:
1990

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix
NAMUR

Institut d'Informatique

Logiciel de simulation d'un jeu de marionnettes pour enfants infirmes moteurs cérébraux.

Laurence Brousmiche
Xavier Gillet

Promoteurs: M. Noirhomme-Fraiture
M. Mercier

Mémoire présenté dans le but de l'obtention du titre de
Licencié et Maître en Informatique
1989 - 1990

Nous tenons à remercier Madame Noirhomme et Monsieur Mercier pour leur aide et les conseils prodigués tout au long de l'élaboration de ce mémoire.

Nous remercions particulièrement Mesdames Truscelli, de Barbot et Charrière pour le merveilleux accueil qu'elles nous ont réservé et leur précieuse collaboration à ce travail. Nous souhaitons beaucoup de plaisir aux enfants du service IMC du CHU de Kremlin-Bicêtre avec le logiciel OrdiThéâtre. Un grand merci à tout le personnel du service, aux enfants et spécialement à Cécile, Perrine et Benoît.

Nous félicitons A. de Theux pour son excellent travail et nous lui exprimons notre gratitude.

Nous présentons nos remerciements à Anne, Marie-Chantal, Martine, Odette, Jacques et Jean-Pierre du département de psychologie, pour leur aide et leur support dans notre travail.

Enfin, nous remercions Madame Deroose ainsi que toutes les personnes ayant collaboré de près ou de loin à la réalisation de ce mémoire.

Abstract.

Nous avons réalisé, dans le cadre de ce mémoire, un logiciel de simulation d'un jeu de marionnettes. Ce logiciel, appelé OrdiThéâtre, a été conçu pour des enfants infirmes moteurs cérébraux, âgés de 6 à 12 ans, dont les capacités motrices ne leur permettent pas d'utiliser un véritable théâtre. Il est destiné à fournir un moyen d'expression et de communication à ces enfants qui en sont souvent privés. Cet ouvrage présente OrdiThéâtre, ses buts, la méthodologie suivie, son architecture et ses spécifications ainsi qu'une analyse détaillée de son interface. Nous exposons les critères généraux de conception d'une interface ainsi que les adaptations matérielles et logicielles à apporter à un tel travail en vue de l'adapter aux diverses caractéristiques de cette population.

We made, within the context of this thesis, a puppets game simulation software. This software, named OrdiThéâtre, has been conceived for physically handicapped children, from 6 to 12 years old, whose motors abilities don't allow them to use a real puppets theater. It intends to provide an expression and communication mean to these children who are often deprived of it. This work presents OrdiThéâtre, its goals, the followed methodology, its structure and its specifications and also a detailed analysis of its interface. We show the generals criteria of interface conception and the hardware and sofware adaptations to bring to a work like this, in view to adapt it to the various characteristics of this population.

Table des matières.

Introduction.

Les personnes handicapées physiques évoluent dans un environnement qui leur est souvent inadapté. De plus, l'impossibilité d'utiliser les modes habituels de communication comme la parole, le toucher, les expressions faciales et gestuelles, rend plus difficile encore leur intégration dans cet environnement. Cette absence de moyens de communication réduit les possibilités d'interaction du handicapé avec son entourage. Cela diminue son champ d'activités et conduit souvent à son isolement. Privé d'autonomie, le geste le plus banal l'oblige à requérir l'assistance d'un tiers. Pour lire un livre, par exemple, il faut pouvoir en tourner les pages. Comment faire pour allumer ou éteindre la lumière, pour ouvrir ou fermer une porte, une fenêtre, pour prendre un objet quelconque lorsque l'on a ses membres paralysés ? Il n'est pas toujours facile ,pour une personne valide, de réaliser l'importance de certains mouvements, certains gestes dans la vie de tous les jours. Se servir un verre d'eau et le boire est hors de portée d'une personne ne pouvant contrôler l'usage de ses bras, de même que de manger une pomme ou une orange. Tous ces gestes nécessitent l'aide d'une autre personne. Mais encore faut-il pouvoir communiquer avec elle. Il faut lui exprimer ses désirs, ses intentions, ses envies. Comment se faire comprendre clairement lorsque l'on ne dispose ni du langage gestuel, ni du langage oral, ni de la possibilité de prendre des objets ? Ce double problème de la réalisation d'une action et de la communication avec une tierce personne augmente la dépendance du handicapé physique vis-à-vis de son entourage. C'est cette dépendance qu'il faut essayer de réduire. Une aide considérable peut être apportée par la technologie pour pallier le plus possible aux déficiences de chacun.

La technologie peut augmenter le degré d'ouverture du handicapé à son entourage en lui fournissant des moyens de communication, d'expression et aussi des possibilités de contrôle de son environnement.

Son rôle est d'essayer de minimiser le handicap en fournissant des substituts aux fonctions physiques et sensorielles faisant défaut, et en optimisant l'utilisation des capacités existantes. Il faut développer les capacités de la personne handicapée et trouver des moyens de contrôle les utilisant, c'est-à-dire déplacer le mode de communication des fonctions déficientes aux fonctions efficientes. La technologie permet d'augmenter le degré d'autonomie des handicapés et ainsi leur permet d'acquérir une indépendance croissante.

Les exemples d'aides par la technologie et, en particulier par l'informatique, foisonnent. Des enfants se révèlent capables d'écrire, de dessiner, de compter et même de faire des dictées grâce à des appareils spécialisés. Des tétraplégiques, totalement immobilisés, commandent leur environnement à la voix, pour appeler au secours, demander un numéro de téléphone, ouvrir ou fermer une porte, lire un livre. Certains peuvent ainsi acquérir une autonomie de 70 % grâce à leur ordinateur. Ils peuvent aussi retrouver une activité professionnelle. En effet, l'introduction de l'ordinateur dans les entreprises diminue les emplois qui exigent un effort manuel. De nombreuses professions sont maintenant caractérisées par un travail sédentaire demandant peu d'effort physique et employant plus les capacités cognitives de la personne.

Un exemple frappant n'est-il pas celui de Stephen Hawking, infirme moteur, considéré par les scientifiques comme un des esprits les plus brillants depuis Einstein. Il déclare dans l'introduction de son bestseller " A brief history of time " : "I have had a lot of help with this book from Brian Whitt, one of my students. I caught pneumonia in 1985, after I had written the first draft. I had to have a tracheostomy operation which removed my ability to speak, and made it almost impossible for me to communicate. I thought I would be unable to finish it. However, Brian not helped me revised it, he also got me using a communications program called Living Center which was donated to me by Walt Woltosz, of Words Plus Inc., in Sunnyvale, California. With this, I can both write books and papers, and speak to people using a speech synthesizer donated by Speech Plus, also of Sunnyvale, California. The synthesizer and a small personal computer were mounted on my wheelchair by David Mason. This system has made all the difference : in fact, I can communicate better now than before I lost my voice."

Vu l'importance de l'informatique pour les personnes handicapées physiques, il est nécessaire de la rendre accessible à une majorité d'entre elles. Nous présenterons dans cet ouvrage deux voies d'adaptation de l'informatique aux infirmes moteurs cérébraux (IMC). La première consiste en la création de matériel spécique permettant l'accès à la machine. La seconde est l'adaptation des logiciels aux divers problèmes de l'utilisation d'un ordinateur par un IMC.

Nous avons conçu le logiciel OrdiThéâtre pour des enfants IMC, âgés de 6 à 12 ans. Ordithéâtre est une simulation d'un jeu de marionnettes sur ordinateur. Il a pour but d'offrir un moyen d'expression, de communication aux enfants qui n'ont pas les possibilités motrices d'utiliser un véritable théâtre. Un jeu de marionnettes a été choisi car le jeu a une importance primordiale pour l'enfant. Il lui apporte du plaisir, c'est un moyen d'apprentissage et d'expression. Il permet à l'enfant d'exprimer ce qu'il vit, de reproduire ce qui, pour lui, est gratifiant, conflictuel ou difficile. L'enfant peut extérioriser ses émotions : ses joies, ses peines, ses angoisses, ses frustrations, ses colères, ses désirs,... et ainsi apprendre à les dominer. Le logiciel se situe dans un cadre, à la fois, pédagogique, ludique et même thérapeutique. Il est décrit en détail dans le chapitre 4.

Dans le chapitre 1, nous présenterons des critères de conception d'une interface homme/machine.

Le chapitre 2 décrira la première voie d'adaptation de l'informatique aux IMC. Différents matériels spécialisés leur permettant l'accès à l'ordinateur seront exposés.

Le chapitre 3 montrera comment s'appliquent les critères vu dans le premier chapitre aux problèmes spécifiques des enfants infirmes moteurs cérébraux. Il s'agira là de la deuxième voie d'adaptation qui est celle de l'adaptation logicielle.

Le logiciel réalisé ainsi que ses buts, la méthodologie suivie, une analyse détaillée de son interface et son architecture seront présentés dans le chapitre 4.

Chapitre 1 : critères de conception d'une interface homme/machine.

Introduction.

L'interface d'une application concerne le dialogue entre l'utilisateur et l'ordinateur. C'est donc à travers elle que l'utilisateur sera en contact avec la machine. Il est, dès lors, indispensable de soigner particulièrement la conception de cette interface pour la rendre attrayante et agréable au vu de la personne employant l'ordinateur. Beaucoup de recherches ont été et sont encore menées dans ce domaine. Il en résulte une importante littérature sur le sujet. Nous exposons dans ce chapitre, un certain nombre de règles et de recommandations glanées un peu partout dans cette littérature. Nous resterons très pratiques en présentant des règles précises et directement applicables.

1. Interface : définition.

Une interface, au sens informatique du terme, est l'ensemble des règles et conventions qui régit la communication entre deux systèmes organisés. [Chernicoff, 1985]

L'interface homme/machine, elle, règle le dialogue entre l'utilisateur et l'ordinateur. Dans le but de faciliter la tâche de l'utilisateur, il est nécessaire de concevoir une interface qui se rapproche le plus possible du mode de pensée et du langage de l'utilisateur.

2. Facteurs humains.

La prise en compte de l'opérateur humain lors de la conception d'un logiciel est primordiale. En effet, la frustration des utilisateurs, conséquence d'une mauvaise interface, risque de provoquer un désintérêt

performante. Il est donc essentiel de s'attarder à satisfaire un certain nombre de critères concernant des facteurs humains. L'importance relative de ces différents critères est déterminée principalement par le type de système à concevoir puisque celui-ci influence considérablement la classe des utilisateurs de l'application.

2.1. Facteurs humains mesurables.

Nous présentons ici, brièvement, cinq facteurs humains mesurables. Ils pourront être utilisés pour évaluer la qualité d'une application interactive.

* Le temps d'apprentissage.

Le temps d'apprentissage est le temps nécessaire à un utilisateur, représentatif de la classe d'utilisateurs visée par le logiciel, pour apprendre et maîtriser le processus de réalisation de la tâche via l'ordinateur. Il est clair que ce temps sera fortement lié à la catégorie de personnes employant l'ordinateur. En effet, l'apprentissage d'un système requiert un effort non négligeable. N'importe quel utilisateur ne sera pas toujours disposé à fournir cet effort pour maîtriser une application informatique.

* Le temps d'exécution.

Le temps d'exécution correspond au temps nécessaire à un utilisateur maîtrisant le logiciel pour réaliser la tâche. L'importance de la prise en compte de ce facteur dépendra fortement du type de tâche à effectuer.

* Le taux d'erreur des utilisateurs.

Le taux d'erreur de l'utilisateur ainsi que le type d'erreurs commises au cours de la réalisation de la tâche sont des éléments importants. Un taux d'erreur trop élevé peut conduire à un mécontentement de l'utilisateur et, dès lors, à un rejet du logiciel.

* La satisfaction subjective.

Ce facteur est sans doute le plus difficile à évaluer mais il n'en est pas moins essentiel. Un utilisateur non satisfait par divers aspects de l'application sera moins enclin à fournir un effort pour la maîtriser et peut-être même pour l'utiliser.

* La rémanence dans le temps.

La rémanence dans le temps est la période pendant laquelle l'utilisateur conserve ses connaissances concernant l'utilisation du logiciel. La fréquence d'utilisation joue certainement un rôle important dans cette période de rémanence.

Comme nous l'avons dit précédemment, il est difficile de satisfaire tous ces critères en même temps. Un temps d'exécution minimal pourra être obtenu par l'emploi d'abbréviations et de raccourcis, mais cela exigera un temps d'apprentissage plus long. Un faible taux d'erreur impliquera généralement un temps d'exécution plus élevé. En effet, l'utilisateur recevra plus d'informations en réponse à ses actions, sera plus guidé et aura moins de libertés dans l'emploi de raccourcis. Lorsque le temps d'apprentissage est important, la satisfaction subjective et la rémanence dans le temps seront primordiales. Un taux d'erreur trop élevé risquera de diminuer cette satisfaction. De plus, celle-ci aura une influence non négligeable sur la fréquence d'utilisation et, par conséquent, sur la rémanence dans le temps.

La détermination des facteurs humains à privilégier est fonction du type de système à concevoir. Celui-ci déterminera en grande partie la classe d'utilisateurs du logiciel.

2.2. Facteurs humains et types de système.

Ben Shneiderman [Shneiderman,1987] distingue quatre catégories de systèmes qui exigeront chacunes de satisfaire différemment les cinq facteurs humains décrits précédemment.

* Les systèmes vitaux.

Les systèmes vitaux comprennent les contrôles du trafic aérien, de réacteur nucléaire, des soins médicaux intensifs, d'opérations chirurgicales, d'opérations de police, militaires,... Dans ces applications, un temps d'apprentissage très long peut être accepté pour obtenir un temps d'exécution performant et surtout un taux d'erreur minimal. La satisfaction subjective est moins importante car les utilisateurs sont motivés et payés. La rémanence se fait par l'emploi fréquent des fonctions ordinaires et par des sessions d'utilisation pour les actions d'urgence.

On comprend aisément qu'un taux d'erreur très faible est primordial dans ce type d'applications : aucune erreur n'est permise lorsqu'il s'agit de soins médicaux intensifs ou du contrôle d'un réacteur nucléaire. De plus, un temps d'exécution performant est essentiel pour faire face à des situations d'urgence. Ces deux critères seront donc ceux dont il faudra le plus tenir compte lors de la conception de l'interface.

* Les systèmes à utilisations industrielles ou commerciales.

Cette catégorie reprend les systèmes informatiques des banques, des assurances, les gestions d'inventaires, des cartes de crédits, des réservations dans les compagnies aériennes, dans les hôtels, les terminaux dans les points de vente,... Ici, un temps d'apprentissage minimal est visé,

ainsi qu'un faible taux d'erreur. Le volume des transactions dans ces systèmes est tel qu'un temps d'exécution réduit est exigé. Par contre, la satisfaction subjective de l'utilisateur est moins importante et, de nouveau, la rémanence est obtenue par une utilisation fréquente.

Shneiderman cite en exemple une enquête effectuée en 1982 par une chaine de motels, qui a montré qu'une réduction d'une seconde dans le temps moyen par réservation, qui était de 150 secondes, économiserait 40 000 $ par an. L'accent est donc porté sur le temps d'exécution. Un apprentissage facile et rapide est recommandé vu le coût élevé que cela occasionne pour l'entreprise. Ces facteurs seront prioritaires par rapport à la satisfaction subjective de l'utilisateur.

* Les applications bureautiques, domestiques et ludiques.

Il s'agit des logiciels de jeu, de traitement de textes, de courrier électronique, de conférence assistée par ordinateur, et de logiciels de gestion. Pour ces systèmes, la facilité d'apprentissage, un faible taux d'erreur et, principalement, la satisfaction subjective sont les critères à satisfaire en priorité. En effet, si les utilisateurs ne maîtrisent pas rapidement le système et s'ils n'en retirent aucune satisfaction, ils se tourneront vers d'autres matériels. Mais il n'est pas facile pour le concepteur de ce type d'applications de créer un logiciel qui plaise à tous. Les débutants préféreront un système simple quitte à augmenter le temps d'exécution. Les utilisateurs expérimentés, eux, désireront un temps minimal tout en disposant d'un maximum de fonctionnalités. Une approche comprenant différents niveaux d'utilisation devra donc être envisagée.

* Les systèmes d'exploration, les systèmes créatifs et les systèmes experts.

Une partie de plus en plus importante des applications informatiques concerne le support d'activités intellectuelles et créatives. Les encyclopédies électroniques, les bases de données, la formation d'hypothèses statistiques, les aides à la décision et la représentation

graphique de résultats de simulation scientifique sont des exemples de systèmes d'exploration. Les applications créatives incluent les systèmes d'aide à la conception et les logiciels de créations musicales. Les systèmes experts s'appliquent aux domaines de diagnostic médical, de prospection pétrolière, de manoeuvre de satellites et de conseils militaires. Les utilisateurs de ces systèmes sont experts dans leurs domaines mais peuvent être débutants face à un ordinateur. Ils sont motivés mais attendent aussi beaucoup de l'application développée. Il est difficile de déterminer quels sont les facteurs humains dont il faut tenir compte principalement car ces domaines d'applications sont encore sujets à énormément de recherches, ce qui complique la caractérisation des utilisateurs de ces systèmes.

Outre les facteurs humains, le type de système exerce aussi une influence sur deux éléments importants dans la création d'une interface : les utilisateurs potentiels et la tâche à réaliser. Leur analyse sera nécessaire avant toute prise de décision concernant le style d'interface à développer.

3. Critères de conception d'une interface.

Nous présentons ici un certain nombre de critères et de recommandations directement applicables lors de la conception d'une interface. En premier lieu seront décrits deux pré-requis de la conception. Ensuite, nous exposerons différents styles de dialogue ainsi que le problème de la structuration de ce dialogue. Nous présenterons, par après, huit règles d'or que le concepteur devra respecter dans un souci de qualité de l'interface. Enfin, quelques conseils concernant la création des divers écans d'un logiciel ne seront pas inutiles et aideront, nous l'espérons, à la réalisation d'une interface de bonne qualité et surtout attrayante.

3.1. Pré-requis de la conception.

L'analyse des utilisateurs potentiels et de la tâche à réaliser est une étape primordiale pour débuter dans la conception d'une interface. En effet, ces deux composants sont les déterminants du style d'interaction à mettre en oeuvre. Ils sont essentiels lors de la prise en compte des différents critères de conception. Les diverses alternatives seront évaluées en fonction de leur analyse.

3.1.1. L'analyse des utilisateurs potentiels.

L'énorme diversité des capacités physiques et intellectuelles, des formations, des motivations et des personnalités rend très difficile la tâche du concepteur. Il est impossible de satisfaire toutes les catégories d'utilisateurs en même temps. C'est pourquoi la caractérisation précise des utilisateurs potentiels est une étape essentielle à laquelle il ne faut pas hésiter à consacrer du temps. C'est cette caractérisation qui déterminera le style d'interaction le mieux adapté.

Un premier élément à analyser sont les caractéristiques physiques des utilisateurs : les possibilités d'accès à la machine, l'âge, le sexe, les possibilités visuelles et auditives. L'interface diffère selon qu'elle s'adresse à un enfant ou à un adulte, à un homme ou à une femme, à une personne valide ou à une personne ayant des difficultés de maîtrise de ses membres. L'utilisateur emploie-t-il le clavier, la souris, le joystick ou simplement, un unique bouton. Cette question est primordiale car il faut adapter l'interface au mode d'accès à l'ordinateur. De même, les possibilités visuelles et auditives sont déterminantes : peut-on employer le son comme mode de communication entre la personne et la machine ; certains éléments doivent-ils être agrandis pour pallier à des déficiences visuelles ; l'emploi de couleurs est-il conseillé ? Toutes les réponses à ces questions sont nécessaires pour entrer dans la conception détaillée de l'interface. Un aspect dynamique doit aussi être envisagé dans des facteurs comme la capacité de voir bouger un objet sur l'écran ou la vitesse d'utilisation du mode d'entrée (par exemple, la rapidité pour presser les touches du clavier).

Un deuxième aspect des utilisateurs potentiels sont leurs capacités intellectuelles : savent-ils lire, écrire, compter, reconnaître les formes et les couleurs; distinguent-ils le haut, le bas, la droite et la gauche ? Il s'agit en fait de définir quel est le référentiel des utilisateurs.

De même, les formations, les motivations et les personnalités sont à prendre en compte. L'utilisateur est-il habitué à travailler avec un ordinateur ? Est-il motivé à l'idée d'utiliser l'application ou cela lui est-il imposé ?

Une difficulté supplémentaire lors de l'analyse des diverses caractéristiques de l'utilisateur est constituée par le fait que celles-ci évoluent constamment. Un utilisateur débutant deviendra, au fur et à mesure de l'emploi du logiciel, expérimenté. Dès lors, il est nécessaire de prévoir plusieurs niveaux d'interface correspondants aux divers niveaux d'expériences.

Toutes ces informations sont destinées à connaître l'utilisateur et ses besoins. Elles peuvent être récoltées par observation de celui-ci et par des interviews. Cela favorise son implication dès le début de la conception de l'interface, ce qui est une nécessité et un atout important pour l'obtention d'un bon résultat. Des lacunes dans ces connaissances risquent de produire un système inadapté et, par conséquent, inutilisé. Beaucoup de temps et d'efforts ont alors été consacré pour un moins bon résultat que celui escompté. Pour éviter une telle déception, due à la création d'un système performant au niveau de la réalisation de la tâche mais inadapté à l'utilisateur, il est de première importance de procéder à une analyse précise de ce dernier.

3.1.2. L'analyse de la tâche.

L'analyse de la tâche consiste en la description et la décomposition de ses fonctionnalités. Il est nécessaire en premier lieu de séparer les fonctionnalités du dialogue. Ces deux éléments doivent être autonomes et indépendants l'un de l'autre. Les fonctionnalités de l'application s'occupent du "quoi", de ce qu'il faut réaliser tandis que le dialogue

s'occupe du "comment" c'est-à-dire de la manière dont s'effectue l'échange d'information entre l'utilisateur et la machine. Il s'agit donc de l'interface entre l'homme et la machine. Ceci peut être modélisé de la façon suivante (figure 4.3.1.):

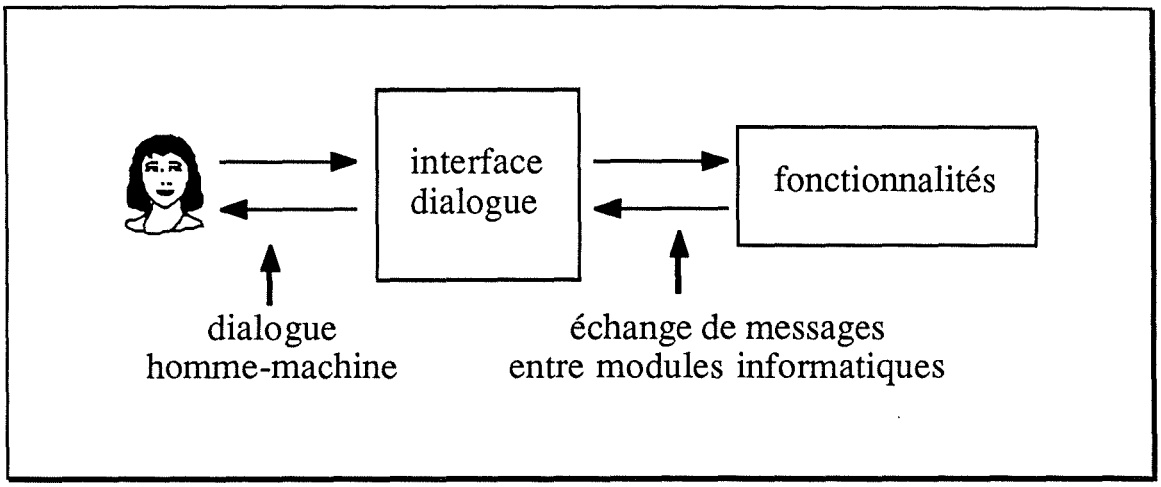


Figure 1.3.1.: la séparation des fonctionnalités et du dialogue.
[Noirhomme,Machgeels, 1990]

La communication entre le système et l'utilisateur est régie par le dialogue. L'utilisateur n'intervient pas dans l'échange d'information entre le module du dialogue et celui des fonctionnalités. L'indépendance des fonctionnalités et du dialogue est primordiale pour assurer la flexibilité et la qualité de l'interface. Sans cette autonomie, la façon dont le dialogue est construit est dirigée par la programmation des fonctionnalités. Or les problèmes se posant dans l'une et l'autre partie sont très différents et requièrent des solutions différentes. Le module des fonctionnalités a besoin de données et il ne se préoccupe pas de la façon dont ces données sont introduites par l'utilisateur. Le dialogue, lui, essaie de satisfaire au maximum les demandes de l'utilisateur. Une même fonctionnalité peut, par exemple, être assurée par divers dialogues en vue de s'adapter aux caractéristiques des différents utilisateurs. Pour résoudre de façon satisfaisante aussi bien les problèmes liés aux fonctionnalités que les problèmes concernant le dialogue avec l'utilisateur, il est nécessaire que ces deux éléments soient totalement indépendants.

L'analyse de la tâche concerne la description et la décomposition des fonctionnalités. Une fonctionnalité met en oeuvre les éléments suivants: la réception d'information (messages); la consultation d'information déjà existante; la mise en oeuvre de règles de traitement et de transformation de ces informations; la mise à jour et la création d'information et l'émission d'information (messages).

Exemple (figure 1.3.2.):

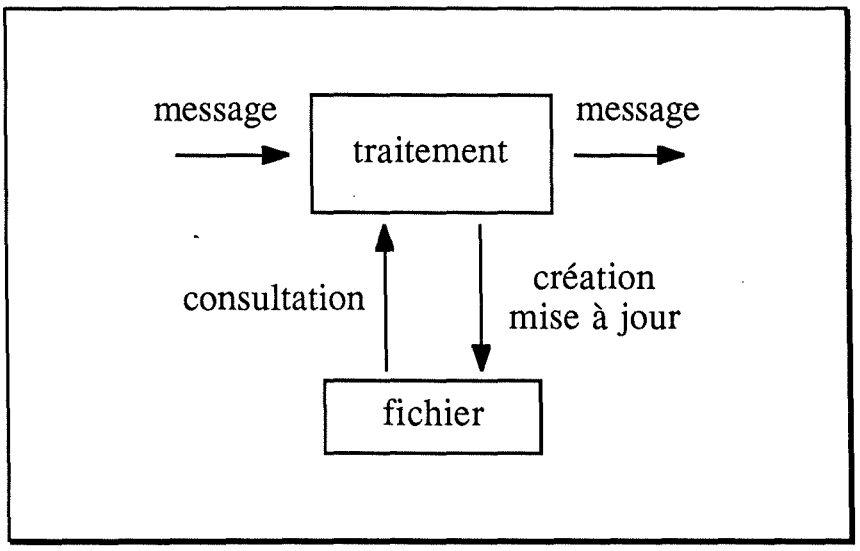


Figure 1.3.2.: exemple de description d'une fonctionnalité.
[Noirhomme,Machgeels, 1990]

Ensuite, il est utile de décomposer les grandes fonctionnalités en fonctionnalités moins complexes car celles-ci sont plus faciles à maîtriser (voir figure 1.3.3.).

Les fonctionnalités définies, il reste à procéder à l'analyse du dialogue.

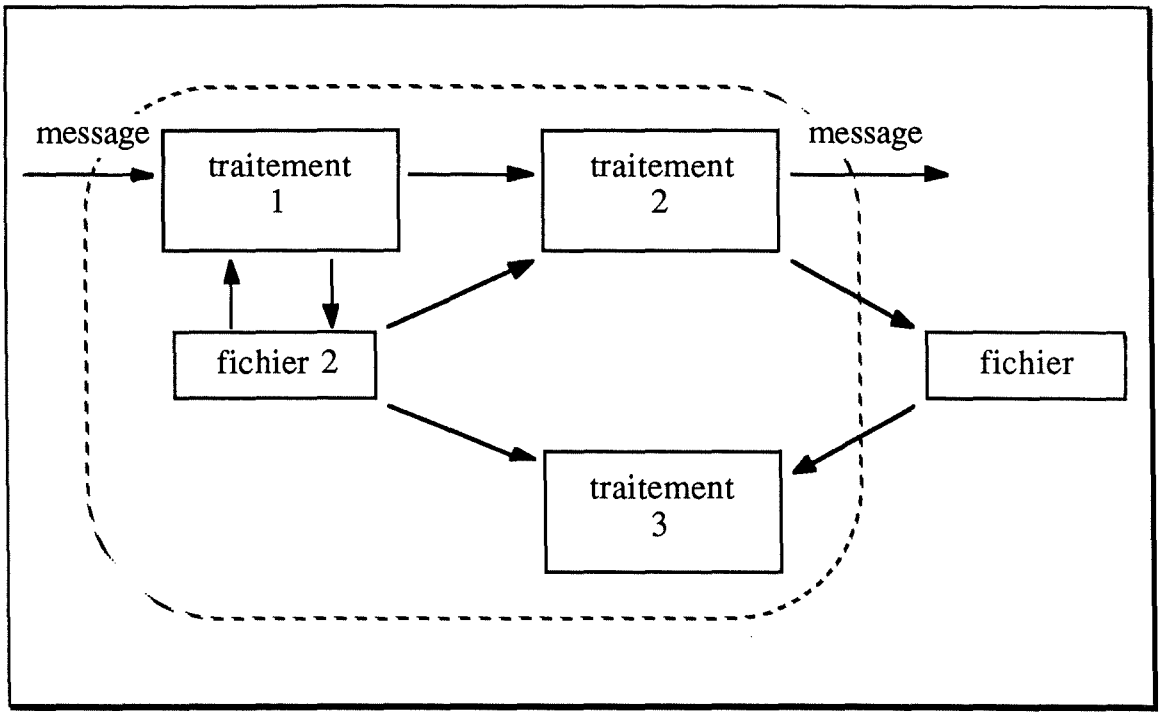


Figure 1.3.3.: décomposition des fonctionnalités.
[Noirhomme,Machgeels, 1990]

3.2. Le dialogue.

Voyons, en premier lieu, les différents styles d'interaction possibles. La meilleure solution à une situation donnée étant rarement homogène, le recours à plusieurs styles est généralement préférable. En effet, il est fréquent d'avoir pour une même application plusieurs profils différents d'utilisateurs. Dans ce cas, il faut envisager divers styles de dialogue. Cependant, leur nombre doit rester limité pour ne surcharger l'utilisateur.

* Le formulaire.

Lorsque l'entrée de données est nécessaire, le style formulaire semble approprié. L'utilisateur dirige un curseur à travers des champs dans lesquels des données doivent être introduites. Pour cela, il doit comprendre la désignation du champ, connaître les valeurs acceptées et la méthode d'introduction des données, ainsi qu'être capable de répondre

aux messages d'erreurs en corrigeant ces dernières. Ceci ne nécessitera généralement qu'un faible apprentissage. Ce type d'interaction est surtout recommandé lorsque la tâche non informatisée requérait également le remplissage d'un formulaire. Dans un tel cas, ce dernier peut être reproduit tel quel à l'écran.

Le formulaire a pour avantages d'être simple, de ne nécessiter qu'un court apprentissage et de fournir beaucoup d'indications. Par contre, l'initiative de l'utilisateur est faible. Il est aussi à remarquer que le remplissage du formulaire demande l'utilisation du clavier. Il faut dès lors s'assurer que l'utilisateur est capable de maîtriser ce mode d'entrée.

* Les langages de commande.

Le langage de commande est un style d'interaction qui laisse l'initiative à l'utilisateur. Celui-ci, après avoir appris la syntaxe du langage, peut rapidement effectuer des opérations complexes et arriver, ainsi, à un certain contrôle de l'application. Ce sentiment de maîtrise augmente sa satisfaction. Malheureusement, le taux d'erreur est généralement élevé car l'utilisateur n'est pas fortement guidé dans son travail. De plus, il est difficile de lui offrir une bonne gestion des erreurs vu la diversité des fautes possibles. Un apprentissage est nécessaire et la période de rémanence est souvent faible car l'utilisateur doit se remémorer les notations.

La forme des différents langages de commande est très variable. Ceux-ci peuvent se résumer à de simples commandes, ou bien avoir une syntaxe complexe. Ils peuvent comprendre quelques opérations ou à l'inverse en contenir un nombre très élevé. Les commandes sont, ou ne sont pas, organisées en une structure hiérarchique. Les abbréviations ne sont pas toujours utilisées. La possibilité pour l'utilisateur de définir des macro-commandes est un attrait important de ce style d'interface. Cela lui permet de réaliser plusieurs opérations avec une seule commande. Celle-ci peut consister en un seul mot ou caractère, ou peut nécessiter des arguments ou des options. Dans le cas d'arguments, il est important que leur ordre soit similaire d'une commande à l'autre pour faciliter leur mémorisation.

Le langage de commande est surtout recommandé pour des utilisateurs expérimentés qui apprécieront d'être les initiateurs du dialogue plutôt que d'être guidés par la machine. De plus, la possibilité d'employer des macro-commandes leur permet une réalisation rapide de la tâche. L'inconvénient majeur de ce type d'interaction est certainement son taux d'erreur élevé. Cependant, un usage fréquent de l'application le diminuera et favorisera la période de rémanence du langage dans la mémoire de l'utilisateur.

* Le langage naturel.

Le langage naturel est souvent considéré comme étant une forme complexe du langage de commande. L'utilisateur n'a pas besoin d'apprendre une syntaxe particulière car il emploie un langage familier (comme le français ou l'anglais) pour donner ses instructions. Ce style d'interaction a pour avantage d'être naturel et rassurant. Cependant, il nécessite souvent un dialogue de clarification vu l'ambiguïté des langues naturelles. De plus, ce n'est jamais réellement une langue naturelle mais plutôt un sous-ensemble restreint de celle-ci.

* Les menus.

Dans ce style d'interface, l'utilisateur lit une série d'options qui lui sont proposées et sélectionne celle qui est appropriée à la tâche qu'il veut réaliser. Lorsque les choix sont bien compréhensibles et distincts, il peut rapidement accomplir ce qu'il veut sans que cela ne lui demande un grand apprentissage, ni des efforts de mémorisation. De plus, il est guidé dans son travail, ce qui diminue son risque d'erreurs. La réalisation complète de la tâche nécessitera généralement un parcours à travers différents menus. Ceux-ci peuvent être organisés de diverses manières (séquence, arborescence, multi-arborescence et réseau de menus).

Il est important de présenter les différentes options d'un menu d'une manière logique. Plusieurs possibilités s'offrent au concepteur : les grouper par ordre chronologique ou alphabétique; présenter les plus

utilisées en premier lieu ou les plus importantes;... Il est nécessaire d'employer une terminologie familière et constante et de bien distinguer les divers choix.

Ce type d'interface est approprié à l'utilisateur débutant ou occasionnel car il est fortement guidé dans la réalisation de sa tâche. De plus, l'emploi de menus ne nécessite qu'un faible apprentissage et peu de mémorisation. Par contre, pour un utilisateur expérimenté, ce style d'interaction peut se révéler trop lent.

* La manipulation directe.

L'utilisateur sélectionne directement des représentations visuelles d'objets ou d'actions en déplaçant un curseur sur l'écran. Ce type d'interface ne requiert qu'un faible apprentissage et la période de rémanence est grande. Le résultat d'une action est rapide et immédiatement visible. L'utilisateur se sent maître de l'application car l'initiative vient de lui.

Il apparaît plus facile de manipuler et de retenir des représentations visuelles que textuelles ou numériques. Elles sont généralement désignées sous le terme d'icônes. Ces icônes doivent être reconnues rapidement et être facilement mémorisables. Elles ont d'autant plus de chances d'avoir ces qualités si elles sont une représentation réaliste et sont très distinctes les unes des autres. Avant d'utiliser une icône, il faut s'assurer qu'elle est comprise de la population cible, ce qui éliminera les risques de confusion.

Les différents styles d'interaction présentés, il reste à développer le problème de la structuration du dialogue. Plusieurs niveaux de dialogue sont définis. Le niveau global consiste en la description d'enchaînement d'étapes, celles-ci étant décrites en détails ailleurs. Le niveau local consiste en la description fine d'une étape du niveau global. En fait, tout est relatif c'est-à-dire qu'un dialogue peut être à la fois un local d'un ou de plusieurs autres et le global d'autres. Par exemple, un dialogue peut être local par rapport à l'application et global par rapport à une option

précise de cette application. Il existe aussi deux types d'enchaînement possibles: séquentiel et asynchrone. Dans le cas de l'enchaînement séquentiel, le dialogue est entièrement piloté par l'utilisateur c'est-à-dire que celui-ci fournit les informations au moment où elles lui sont demandées. L'enchaînement asynchrone offre l'initiative à l'utilisateur dans le déroulement du dialogue. Il est généralement plus adapté à un utilisateur expérimenté.

La structuration du dialogue consiste donc en la définition des différents niveaux de dialogue et du type d'enchaînement le plus approprié.

3.3. Les règles d'or de la conception.

Cette section présente 8 principes de conception qui sont applicables dans les systèmes interactifs. Ils sont généraux et doivent donc être interprétés et redéfinis en fonction de l'environnement précis auquel on est confronté.

Les principes exposés sont ceux de Ben Shneiderman [Shneiderman,1987], qu'il intitule les règles d'or de la conception.

L'uniformité requiert les mêmes séquences d'actions dans des situations similaires, une terminologie constante et des commandes uniformes. Des exceptions peuvent être faites pour attirer l'attention de l'utilisateur lors de commandes dangereuses comme la suppression ou la modification. Il est important d'employer une terminologie compatible avec le vocabulaire de l'utilisateur, dans un environnement constant.

L'emploi de raccourcis doit être permis aux utilisateurs expérimentés. Ceux-ci préfèrent, en effet, exécuter un plus petit nombre d'actions agrégées pour augmenter leur vitesse de réalisation de la tâche. L'utilisation d'abbréviations, de touches particulières et de macro-commandes est appréciée par cette classe d'utilisateurs, ainsi qu'un temps de réponse relativement court.

Un feedback informatif est requis pour chaque action de l'utilisateur. Pour des petites actions fréquentes, la réponse peut être très

limitée. Mais pour des actions importantes ou plus rares, elle doit être plus élargie et explicite.

L'organisation de séquences d'actions comprenant un début, un milieu et une fin, est nécessaire. L'information reçue en retour à la fin d'une séquence procure à l'utilisateur la satisfaction d'avoir accompli quelque chose et l'informe que la voie est libre pour entamer une nouvelle séquence.

Une gestion simple des erreurs est recommandée. Le système doit être conçu de telle sorte que l'utilisateur ne puisse pas commettre d'erreur grave. Une gestion simple nécessite, par exemple, que l'utilisateur ne soit pas obligé de retaper toute la commande erronée mais qu'il puisse corriger uniquement la partie fautive.

Le retour en arrière doit être permis le plus souvent possible. Cela évite à l'utilisateur d'être anxieux : il sait en effet qu'en cas d'erreur, il a la possibilité d'annuler son action et de revenir sur ses pas. Il est alors encouragé à explorer les options qu'il connaît moins et il augmente ainsi sa maîtrise de l'application.

Le contrôle explicite de l'application doit être donné à l'utilisateur. Cela signifie que même si c'est le logiciel qui a le contrôle, l'interface doit apparaître comme étant sous la direction de l'utilisateur. C'est uniquement en réponse d'actions explicites de ce dernier que le logiciel devra exécuter des opérations. En résumé, l'utilisateur doit être l'initiateur de l'action.

La charge de mémoire à court terme doit être réduite car la probabilité d'erreur humaine augmente dans les situations à charge élevée. C'est pourquoi un maximum d'informations doit être fourni à l'utilisateur pour qu'il sache, à tout moment, où il en est.

3.4. La création d'écrans.

Si elle est bien conçue, l'organisation des informations sur les divers écrans du logiciel constitue un attrait important pour l'utilisateur. C'est pourquoi il est indispensable de créer avec soin ces différents

écrans. Pour aider le concepteur dans ce travail, on trouve, dans la littérature, des règles élémentaires et des recommandations.

En premier lieu, il est nécessaire d'effectuer des regroupements logiques d'informations et il n'est pas inutile de prévoir un renforcement visuel de ces regroupements. Il est aussi préférable de conserver une homogénéité de présentation d'un écran à l'autre. Une attention particulière doit être portée sur les informations à présenter : il est déconseillé de surcharger l'écran avec des données inutiles. Quant à la disposition de ces données, il vaut mieux privilégier une organisation haut-bas plutôt que gauche-droite. Des espacements suffisants sont requis ainsi qu'un alignement des informations.

Il existe aussi un certain nombre de techniques pour attirer l'attention de l'utilisateur sur des éléments précis. Par exemples, l'emploi de plusieurs degrés d'intensité, de diverses tailles de caractères, du clignotement et de l'affichage en inverse, de la couleur et des sons. Un marquage particulier peut être réalisé en soulignant une donnée, en la positionnant dans un cadre ou en l'indiquant au moyen d'une flèche. Une grande prudence est recommandée dans l'utilisation de ces techniques. En effet, leur emploi de manière excessive risque de produire un écran encombré et, par conséquent, risque de provoquer une certaine confusion de l'utilisateur. Les débutants ont besoin, pour les guider, d'écrans simples et organisés logiquement. Pour les personnes expérimentées, une mise en évidence subtile des informations importantes est suffisante.

Conclusion.

Le logiciel que nous avons créé dans le cadre de ce mémoire s'adresse à des enfants infirmes moteurs cérébraux. Pour ce type d'utilisateurs, une interface particulière est nécessaire. Vu leurs difficultés d'accès à la machine, il est primordial que l'interface leur soit adaptée et qu'elle soit suffisamment attrayante pour qu'ils ne jugent pas inutiles leurs efforts dans l'utilisation de la machine. C'est pourquoi, en tant que concepteurs, nous avons essayé d'appliquer et d'adapter au maximum les règles et recommandations énoncées dans ce chapitre. Les deux chapitres

suivants présenteront respectivement des adaptations matérielles et logicielles de l'informatique aux infirmes moteurs cérébraux.

Chapitre 2 : Première voie d'adaptation de l'informatique aux IMC : l'utilisation de matériel spécifique.

Introduction.

Les innovations dans la technologie des interfaces sont telles que les personnes les plus gravement handicapées sur le plan physique peuvent actuellement utiliser un ordinateur. Il y a quelques années, l'idée qu'un tétraplégique puisse contrôler seul une telle machine semblait relever de l'utopie. Aujourd'hui, une multitude de systèmes d'entrée-sortie spécialisés permettent aux handicapés d'accéder aux ordinateurs et à leurs innombrables possibilités.

Afin de cerner les problèmes que pose l'utilisation de l'informatique par les infirmes moteurs cérébraux (IMC), nous proposerons dans ce chapitre une définition de l'infirmité motrice d'origine cérébrale. Ensuite, nous passerons en revue les principales techniques adaptées aux différents troubles rencontrés.

1. Définition de l'Infirmité Motrice d'origine Cérébrale.

Afin de dégager les adaptations matérielles nécessaires à l'utilisation de l'informatique par les IMC, il nous semble utile de décrire brièvement ce que l'on entend par IMC : "Nous appellerons IMC celui qui a été atteint avant, pendant ou peu de temps après sa naissance, d'une anomalie non évolutive et non curable des tissus cérébraux, se manifestant entre autres par des troubles moteurs. L'Infirmité Motrice d'origine Cérébrale est donc un état pathologique et non une maladie." [Robaye, 1975].

On distingue plusieurs types de troubles moteurs dont les deux plus importants semblent être la spasticité et l'athétose. Dans la spasticité, les muscles sont raides et hyperexcitables. Lorsqu'un mouvement est

ébauché, les muscles qui devraient se relâcher n'y parviennent pas, ce qui fait que le mouvement s'arrête à mi-chemin ou est paralysé. Un athétosique, par contre, est agité de mouvements spasmodiques involontaires, incontrôlés et non coordonnés qui rendent l'action maladroite et difficile.

Ces troubles sont de gravité variable et se situent à différents niveaux : membres inférieurs et supérieurs, tête, face, ... En conséquence, nous nous trouvons devant des personnes dont les difficultés sont multiples et variées. Cela peut aller de problèmes dans l'utilisation des bras ou des jambes jusqu'à la paralysie quasi-totale, en passant par des troubles de la parole dus à des problèmes moteurs au niveau de la face empêchant de former les sons du langage. A ceci peuvent s'ajouter des troubles dits *associés*, tels que des troubles de la vue ou de l'audition à des degrés divers, des problèmes d'organisation spatiale ou des carences au niveau du schéma corporel. Le fonctionnement intellectuel global de l'IMC peut varier entre la débilité mentale profonde et l'intelligence supérieure et brillante.

Vu la diversité des troubles, le matériel nécessaire à l'utilisation de l'ordinateur par des IMC est spécialisé et étendu. Le choix et la combinaison des différents systèmes existants doivent être étudiés en fonction de chaque utilisateur afin d'offrir un maximum de possibilités, de facilité d'emploi et de confort d'utilisation. Il n'existe donc pas de compromis apte à satisfaire l'ensemble de la population IMC.

2. Solutions techniques et applications.

Un concept clé de l'utilisation de l'informatique par des handicapés moteurs est celui d'*accès transparent*, qui devrait leur permettre d'utiliser n'importe quelle application existante quel que soit le matériel d'entrée-sortie utilisé. Nous allons passer en revue les principaux systèmes d'accès à l'ordinateur utilisés par les IMC et tenter d'établir les conditions dans lesquelles ils sont utilisables.

2.1. Les claviers.

Le clavier reste toujours le principal mode d'entrée de données. Si des études poussées ont contribué à l'amélioration de son ergonomie (taille, inclinaison, espacement et largeur des touches, force d'appui nécessaire...), il n'en reste pas moins mal adapté, voire totalement inutilisable, pour de nombreux handicapés physiques. Les claviers standards devront donc être adaptés à l'aide d'accessoires particuliers ou même remplacés par des claviers spécifiques, conçus pour être utilisés dans des conditions précises.

2.1.1. Adaptations possibles des claviers standards.

La première adaptation consiste à fixer un *cache-clavier* (ou guide-doigts). Il s'agit d'une plaque perforée, généralement en plexiglas, qui se place sur le clavier de manière à ce que les touches ne soient accessibles qu'à travers les trous prévus à cet effet. Cela permet à l'utilisateur de reposer ses mains sur le clavier et rend l'accès à ce dernier plus aisé lorsque les mouvements sont imprécis ou saccadés. Le guide-doigts facilite également le maniement du clavier à l'aide d'une licorne (tige rigide placée sur le front) ou d'une mentonnière.

Certaines touches (SHIFT ou CONTROL par exemple) demandent à être manoeuvrées en même temps que d'autres, ce qui n'est pas possible lorsque l'utilisateur ne dispose que d'une seule main ou travaille avec une licorne. Ce problème peut être réglé par l'utilisation de *bloque-touches*, qui maintiennent appuyées les touches voulues.

Finalement, certains systèmes permettent de supprimer la répétition automatique et autorisent une temporisation du clavier. Les touches doivent alors être enfoncée un certain temps avant d'être acceptées, ce qui réduit le risque de sélection accidentelle.

2.1.2. Les claviers spéciaux.

Si des adaptations très simples rendent possible l'utilisation d'un clavier standard par certaines personnes gravement handicapées, il existe bien des cas où un clavier spécifique s'avère être la seule solution possible. Ces claviers offrent généralement la possibilité de régler le délai de réaction, possèdent des touches spéciales - telles que SHIFT et CONTROL - verrouillables et émettent un retour sonore. Ils se distinguent entre eux par leurs dimensions et leurs caractéristiques ergonomiques.

Les *mini-claviers* sont destinés aux utilisateurs dont les mouvements ne sont précis que s'ils sont de faible amplitude. Les touches sont de taille réduite et sont accolées les unes aux autres. Etant généralement deux fois plus petits que la normale, ces claviers s'adaptent facilement sur une chaise roulante.

Les *claviers agrandis* s'adressent à tous ceux dont les mouvements n'ont pas la précision requise pour un clavier normal. Les touches, très espacées et fortement élargies, s'activent même par des mouvements très désordonnés de la main, du bras ou du menton.

Les *claviers à commande buccale* se présentent sous la forme d'une cible circulaire placée verticalement et sur laquelle sont disposées les touches. Un bâtonnet équilibré, fixé dans un plan perpendiculaire au clavier, peut être dirigé, à l'aide de la bouche ou du menton, sur une touche quelconque. Celle-ci peut alors être actionnée en soufflant légèrement dans l'embout pneumatique du bâtonnet. Ce dispositif peut être employé par des handicapés ne pouvant pas se servir de leurs membres supérieurs, mais ayant un bon contrôle de la tête.

2.1.3. Le système MOD.

Le système de clavier MOD est un dispositif très puissant qui a été développé par le National Research Council of Canada. Il se compose d'un petit ordinateur bon marché utilisé comme ordinateur *front-end,* d'un écran de contrôle et d'un système de sélection (un joystick ou un jeu

d'interrupteurs, par exemple). Le software basé sur le front-end affiche sur son écran de contrôle une représentation graphique d'un clavier et envoie à l'ordinateur hôte le code de la touche sélectionnée par l'utilisateur. L'utilisation d'un ordinateur et d'un moniteur pour simuler le clavier offre une très grande souplesse. En effet, il suffit de changer le dispositif de sélection et le programme du système MOD pour l'adapter à un utilisateur particulier. Grâce à cela, des personnes souffrant de divers lourds handicaps physiques peuvent accéder à la plupart des applications tournant sur un grand nombre d'ordinateurs, sans aucune modification du software.

2.2. Les systèmes de manipulation directe.

Lorsqu'un écran est utilisé pour présenter de l'information, il est souvent utile de pouvoir sélectionner certains éléments. L'approche *manipulation directe* est intéressante car elle évite aux utilisateurs l'apprentissage de commandes, elle réduit les risques de fautes de frappe au clavier et permet de rester attentif à ce qui se passe sur l'écran. Les résultats en sont souvent de meilleures performances, un taux d'erreur moindre, un apprentissage plus facile et une satisfaction plus élevée. Pour un handicapé moteur, un système de manipulation directe bien adapté est parfois le seul moyen possible pour accéder à un ordinateur.

2.2.1. Les tâches réalisables à l'aide de ces systèmes.

Les systèmes de manipulation directe sont applicables dans six types de tâches interactives [Foley et al., 1984]:

1. *Sélection* : l'utilisateur choisit parmi un set d'items. Ceci peut être une sélection traditionnelle dans un menu, l'identification d'un fichier dans un répertoire, ...

2. *Positionnement* : l'utilisateur choisit un point dans un espace à une, deux ou trois dimensions.

3. *Orientation* : l'utilisateur choisit une direction dans un espace à deux ou trois dimensions.

4. *Description d'un chemin* : l'utilisateur effectue une série d'opérations de positionnement et d'orientation. Le chemin peut par exemple représenter une courbe dans un programme de dessin ou une route sur une carte.

5. *Quantification* : l'utilisateur spécifie une valeur numérique, généralement pour fixer certains paramètres tels que l'amplitude d'un son ou le nombre de pages d'un document.

6. *Edition de textes* : l'utilisateur introduit, déplace et édite du texte dans un espace à deux dimensions. Le système de manipulation (souris, joystick ou autre) sert notamment à placer le point d'insertion, sélectionner une partie du texte, indiquer les marges, le type de caractères ou le style employé. En plus de ces fonctions classiques, il permet également d'entrer le texte dans l'ordinateur en sélectionnant les caractères à l'écran, à condition d'utiliser un software particulier affichant un clavier sur une partie du moniteur.

Il est bien entendu possible d'effectuer toutes ces tâches avec un clavier, en tapant par exemple des nombres ou des lettres pour sélectionner et des coordonnées pour positionner. Cependant, des appareils ont été créés pour réaliser ces fonctions plus rapidement et avec moins d'erreurs. Les systèmes développés pour des utilisateurs valides peuvent être employés avec plus ou moins de bonheur par des handicapés moteurs et certains matériels spécifiques ont été créés pour ouvrir les voies de la manipulation directe à un maximum de personnes.

2.2.2. Les matériels classiques et spécifiques.

Le *crayon optique* permet à l'utilisateur de désigner un point directement sur l'écran, ce qui le rend très intéressant. Afin d'éviter les sélections involontaires, la plupart des crayons optiques possèdent un bouton qui doit être enfoncé lorsque le curseur se trouve à la position

désirée. L'*écran tactile* est basé sur le même principe, mais l'élimination de l'intermédiaire entre l'écran et l'utilisateur rend son maniement encore plus simple. Par rapport au crayon optique, il présente une moins grande précision, due à l'épaisseur du doigt humain, mais le fait de ne pas avoir à se saisir d'un objet le rend généralement préférable. Malgré deux inconvénients majeurs, à savoir la fatigue du bras provoquée par le contrôle direct sur un plan vertical d'une part et le problème de la main masquant une partie de l'écran d'autre part, ces deux systèmes sont très bien adaptés à de nombreux IMC. Ils ne requièrent pratiquement aucun apprentissage et ne nécessitent que des gestes très simples d'un seul bras. Lorsque les logiciels présentent de larges zones de contrôles sur l'écran, les mouvements peuvent être relativement imprécis ou légèrement saccadés, surtout si l'on utilise un crayon optique possédant un bouton de confirmation.

La *souris* permet à la main de rester dans une position confortable. Elle permet des déplacements rapides du curseur et un positionnement très précis mais demande une certaine habitude pour être vraiment efficace. Son utilisation est cependant problématique ou impossible pour certains IMC. En effet, en plus de gestes précis de la main, le maniement d'une souris demande une excellente coordination main-yeux, une bonne organisation spatiale et exige un niveau d'abstraction plus élevé que d'autres dispositifs. En plus de ces problèmes principaux, quelques petits inconvénients peuvent être notés: la souris utilise beaucoup de place et est donc difficilement utilisable sur la tablette d'un fauteuil roulant, le câble peut être dérangeant et les longs déplacements du curseur nécessitent de fréquents repositionnements de la souris.

La *trackball* est semblable dans son principe à une souris à l'envers. Les mouvements du curseur sont obtenus en faisant rouler la boule de contrôle avec la main. Sa précision et sa rapidité sont comparables à celles d'une souris mais, contrairement à cette dernière, la trackball ne doit pas être déplacée. Elle permet de contrôler le curseur en ne se servant que d'un seul doigt, ce qui en fait une aide très précieuse pour les personnes qui ne peuvent réaliser que de légers mouvements avec la main.

Le *joystick* ne demande que des mouvements de faible amplitude et permet des changements de direction très aisés. Relativement lent et peu précis lorsqu'il s'agit de guider un curseur vers une destination fixe, ce système est particulièrement efficace pour tracer un chemin ou suivre un objet se déplaçant à l'écran. Sa similitude avec la manette de commande d'un fauteuil roulant électrique le rend familier à certaines personnes lourdement handicapées. Il existe une large gamme de joysticks qui se différencient par l'amplitude des mouvements à effectuer, la force à exercer, la taille et l'épaisseur du manche, le nombre de boutons de commande. Il est donc possible de trouver des joysticks adaptés à différents types de handicaps: certains sont destinés à être manoeuvrés à une main, d'autres avec un ou deux doigts, d'autres encore avec le menton ou la bouche.

La *tablette graphique* consiste en une surface sensitive séparée de l'écran, généralement posée à plat sur la table. Ceci permet une position confortable de la main, mais empêche l'utilisateur de conserver son attention sur l'écran. Un des intérêts principaux de ce système est d'offrir une assez grande surface sur laquelle on peut disposer des dessins, des mots ou autres symboles (des symboles Bliss, par exemple) permettant de figurer un clavier ou des commandes et ce, sans encombrer l'écran.

Le *casque ultrasonique* est composé d'une part, d'un casque léger comportant deux récepteurs d'ultrasons et un contacteur pneumatique et d'autre part, d'un émetteur d'ultrasons. Les déplacements du curseur sont fonction des mouvements de la tête et la validation se fait en soufflant légèrement dans le tube du contacteur. Ce système, très rapide mais relativement peu précis, permet à l'utilisateur de rester attentif à ce qui se passe sur l'écran. Le casque ultrasonique est avant tout destiné aux personnes handicapées ne pouvant faire usage de leurs membres supérieurs, mais ayant un bon contrôle de la tête.

La technologie des *systèmes de détection des mouvements oculaires* offre un très grand potentiel, mais elle est encore extrêmement coûteuse et peu répandue. Lorsque des objets interactifs sont présentés sur un écran, l'utilisateur commence généralement par fixer son attention visuelle sur l'item désiré avant de le sélectionner. Ceci se matérialisant

par un mouvement des yeux vers la "cible", un appareil capable d'analyser les mouvements oculaires semble donc un système de manipulation directe très naturel et très rapide. Dans l'optique d'une utilisation par des IMC, un tel outil permet aux personnes les plus gravement atteintes, dont le regard peut être le seul moyen d'expression et de communication, de contrôler un ordinateur et, via celui-ci, d'avoir une action sur le monde extérieur. Les techniques actuelles posent encore certains problèmes: la précision est quelque peu affectée par la nature saccadée des mouvements oculaires, les mouvements de la tête peuvent dérégler le système (certains appareils n'ont pas cet inconvénient, mais leur prix dépassait les cent mille dollars en 1987 [Ware, Mikaelian, 1987]) et les méthodes de sélection ne sont pas encore totalement satisfaisantes. Dans l'état actuel des choses, la validation peut se faire soit par l'utilisation d'un bouton ou d'un interrupteur, soit par l'immobilisation du regard durant un délai de plus ou moins une demi-seconde. Lorsqu'elle est possible, la solution de l'interrupteur est préférable car plus rapide, plus fiable et risquant moins de provoquer des sélections accidentelles.

2.3. Les interrupteurs.

Il existe un nombre impressionnant d'interrupteurs simples, doubles ou multiples, permettant d'utiliser pratiquement n'importe quel mouvement intentionnel du corps. A côté des simples "boutons" de tailles et de formes multiples, figurent des systèmes variés et sophistiqués tels que languettes flexibles, interrupteurs au mercure activés par balancement, interrupteurs souples, pédales et autres contacteurs pneumatiques, sonores ou photosensibles.

Pour être utilisables, ces contacteurs doivent être intégrés dans le reste de l'installation. Ils peuvent par exemple être branchés directement sur certains appareils afin de changer un système mal adapté tel que le bouton de la souris ou un interrupteur de marche-arrêt. Mais il est également possible de les raccorder à des boîtiers de contrôle ou des cartes d'extension afin de construire des systèmes d'entrée-sortie complets

et performants. Ainsi, des dispositifs tels que la **Magic Box**[1], qui permet de remplacer jusqu'à seize touches du clavier par des interrupteurs quelconques, ou la **Carte Firmware**[2], qui permet notamment l'entrée de données à l'aide d'un seul contacteur (par l'utilisation du code morse ou de techniques de balayage), facilitent l'utilisation de logiciels standards.

2.4. La reconnaissance de la parole et la synthèse vocale.

Les systèmes de reconnaissance de la parole deviennent plus fréquents et offrent un grand potentiel aux personnes handicapées qui ne peuvent pratiquement pas utiliser leurs membres mais ne sont pas atteintes de graves troubles du langage. Ces techniques sont limitées à des applications particulières; il n'est bien entendu pas question de pouvoir tenir une conversation avec un ordinateur...

2.4.1. La reconnaissance de mots.

Des appareils capables de reconnaître des mots prédéfinis, dits par une personne particulière, fonctionnent actuellement avec une fiabilité de l'ordre de 90 à 98 pourcents. Et ce, pour un vocabulaire de 50 à 150 mots [Shneiderman, 1987]. La plupart de ces systèmes sont dépendants de l'utilisateur et ce dernier doit donc "apprendre" sa prononciation à l'ordinateur en répétant l'entièreté du vocabulaire une ou plusieurs fois. Cette technique offre l'avantage important d'être utilisable même par des personnes ayant de légers troubles de la parole et est beaucoup plus fiable que les systèmes indépendants de l'utilisateur, actuellement en cours de développement.

Les applications les plus convaincantes de la reconnaissance de mots sont destinées aux handicapés. Ils peuvent ainsi contrôler des fauteuils roulants, manoeuvrer certains équipements ou accéder à des micro-ordinateurs. En règle générale, cette technique n'est réellement

[1] La Magic Box existe en configuration Apple II GS, Apple Macintosh et Commodore Amiga.
[2] La Carte Firmware existe en configuration Apple II Plus, Apple IIe et Apple II GS.

intéressante que sous certaines conditions particulières, par exemple lorsque les mains ou les yeux de l'utilisateur sont occupés, ou lorsqu'une grande mobilité est requise.

2.4.2. La reconnaissance d'un flux continu de paroles.

Des recherches sont en cours afin de produire des systèmes capables de reconnaître des flux continus de parole, mais beaucoup d'observateurs pensent qu'une version commercialisable d'un tel produit ne sera pas au point avant une bonne dizaine d'années. Les applications que l'on pourrait baser sur cette technique sont nombreuses, tant pour les handicapés que pour les personnes valides. Les espoirs fondés sur la reconnaissance de flux de paroles sont, entre autres, de pouvoir un jour dicter une lettre à un appareil de transcription automatique et de permettre à un ordinateur d'inspecter de longues bandes audio, des programmes radio ou des appels téléphoniques afin d'y retrouver certains mots-clés.

En 1985, Kuzweil Computers Products a annoncé un système expérimental capable de reconnaître des flux de paroles avec un vocabulaire de 3000 mots mais il faut continuer à le tester afin de déterminer ses véritables capacités en situation réaliste [Shneiderman, 1987].

2.4.3. La synthèse vocale.

Des systèmes de synthèse vocale efficaces et bon marché sont actuellement disponibles. Ils sont d'un intérêt évident pour les aveugles et les personnes souffrant de troubles visuels ainsi que pour ceux qui ne peuvent s'exprimer normalement par la parole. Ils ouvrent de nouvelles voies de communication et sont particulièrement utiles à de nombreux IMC. La synthèse vocale peut être utilisée afin de renforcer les messages produits par un ordinateur ou en tant que pièce maîtresse d'un système d'expression et de communication destiné à des personnes lourdement handicapées.

Conclusion.

Les matériels visant à améliorer l'accès a l'ordinateur sont légion, mais aucun des appareils existants ne peut répondre aux besoins de toutes les personnes handicapées. La conception d'une interface adaptée doit reposer sur une étude précise de chaque cas individuel et doit être menée par des gens compétents. Un système bien pensé doit être cohérent et ergonomique; il doit maximiser les performances de l'utilisateur en tirant parti des mouvements les plus efficaces qu'il peut effectuer volontairement. Dans ces conditions, l'informatique devient à la portée des handicapés physiques les plus gravement touchés. Dans les meilleurs des cas, la mise en place de hardware particulier permet l'utilisation de la plupart des logiciels standards. Par contre, lorsque les moyens d'entrée sont trop réduits ou que le software est trop complexe, la conception de logiciels spécifiques s'impose.

Chapitre 3 : Deuxième voie d'adaptation de l'informatique aux IMC : les adaptations logicielles.

Introduction.

Pour permettre l'accès à l'ordinateur aux Infirmes Moteurs Cérébraux, il est essentiel de mettre à leur disposition un matériel spécialisé. Mais ce matériel, à lui seul, n'est pas suffisant. Il est aussi nécessaire de créer des logiciels adaptés aux différentes caractéristiques d'un IMC. Dans le cadre de notre stage à Paris, nous avons travaillé avec des enfants IMC âgés de 6 à 12 ans. Nous avons pu dégager un certain nombre d'observations et ainsi formuler des recommandations quant à la conception d'une interface adaptée à ces enfants. Nous reprendrons, ici, les différents conseils et critères énoncés dans le premier chapitre concernant la conception d'une interface homme/machine, et nous les appliquerons en fonction des diverses caractéristiques de cette population.

1. Facteurs humains.

* Le temps d'apprentissage.

Vu leurs déficiences, ces enfants sont habitués à fournir de nombreux efforts pour arriver à la maîtrise d'un geste, d'un mouvement, d'une action. Ils sont d'autant plus motivés si ce geste ou cette action leur permet d'acquérir ne fut-ce qu'un peu d'indépendance. Ils ne rebuteront donc pas devant l'effort et ne se décourageront pas trop vite pour autant que l'envie et la motivation de maîtriser le logiciel soient suffisamment importantes. Pour cela, il faut que celui-ci soit très attrayant et que les enfants soient autonomes dans l'utilisation du logiciel c'est-à-dire qu'ils aient l'impression de réaliser quelque chose à eux seuls. Il peut donc y avoir un apprentissage mais il ne faut pas oublier que pour des enfants

IMC, cet apprentissage est en quelque sorte doublé. S'il est trop long, les enfants risquent de ne pas percevoir ce que le logiciel peut leur apporter et ils jugeront alors les efforts à fournir comme étant inutiles.

* Le temps d'exécution.

Ce temps n'est pas très important vu la nature de la population. En général, ces enfants passent énormément de temps dans l'utilisation du mode d'entrée. Certains doivent se concentrer et fournir un gros effort pour presser la touche du clavier qu'ils désirent, d'autres mettront un temps considérable pour placer la souris à l'endroit voulu et ensuite, cliquer sans qu'elle ne bouge de place. Ceci implique que, même si le logiciel est performant, le temps d'exécution sera agrandi par les problèmes liés à l'entrée des données.

* Le taux d'erreur.

Etant donné leurs problèmes de précision dans l'utilisation des appareils d'entrée des données, les enfants IMC sont plus enclins à commettre des erreurs. Ils seront facilement découragés si le logiciel ne réagit pas convenablement vis-à-vis de ces erreurs. L'interface doit donc être développée de telle sorte qu'elle évite le plus d'erreurs possible même si cela a pour conséquence de diminuer la puissance du logiciel. L'enfant ne doit pas être continuellement dérangé par des messages lui indiquant qu'il a poussé sur une mauvaise touche ou qu'il a cliqué en dehors de la zone voulue. Dans ce cas, le logiciel doit simplement attendre la bonne valeur sans se soucier des entrées erronées.

* La satisfaction subjective.

Ce critère est primordial. Cette satisfaction peut provenir de nombreux aspects du logiciel. Les IMC peuvent y voir une possibilité d'autonomie, un moyen de jeu, d'apprentissage, de communication... Il est à noter que ces enfants seront sans doute plus vite satisfaits que d'autres

puisqu'ils ont accès à beaucoup moins de choses que des enfants valides. De plus, vu l'effort qu'ils doivent fournir, il est essentiel qu'ils en retirent une grande satisfaction pour qu'ils jugent cet effort utile.

* La rémanence.

Il est important que la rémanence dans le temps soit grande. En effet, il faut éviter de devoir recommencer l'apprentissage trop souvent car celui-ci est parfois long et difficile. L'interface doit donc être conçue de manière à ce que l'utilisation du logiciel soit facilement mémorisable par des enfants IMC.

2. L'analyse des utilisateurs.

L'analyse des utilisateurs constitue une étape essentielle dans la conception d'une interface. Elle est d'autant plus importante lorsqu'il d'agit d'une population particulière comme celle d'enfants IMC. Il est nécessaire qu'elle se situe aussi bien au niveau des capacités physiques qu'intellectuelles ainsi qu'au niveau des motivations.

2.1. Niveau physique.

- la population considérée est constituée d'enfants infirmes moteurs cérébraux, âgés de 6 à 12 ans, garçons et filles.

- les possibilités d'accès à l'ordinateur sont très variées. Certains enfants emploient le clavier, mais ils préféreront la souris lorsqu'ils sont capables de manier l'un et l'autre. Pour les enfants paralysés des membres supérieurs mais possédant une très grande précision avec la tête, le casque ultrasonique sera plus adapté qu'une licorne. Le clavier ne sera donc utile que pour les enfants ne sachant pas utiliser la souris ou tout autre matériel la remplaçant, mais qui peuvent taper au clavier plutôt que d'employer un interrupteur. Une majorité d'IMC utilisera la souris ou un matériel équivalent (joystick, trackball, casque

ultrasonique,...), et une faible minorité travaillera à l'aide d'un simple interrupteur, celui-ci rendant la tâche longue et fastidieuse. Cependant, même s'il s'agit d'une minorité, il faut aussi en tenir compte pour rendre le logiciel accessible à un maximum de ces enfants. En conséquence, pour obtenir une interface adaptée aux IMC, il est essentiel de prévoir plusieurs modes d'entrée des données en parallèle aux divers modes d'accès à l'ordinateur.

- les possibilités visuelles : de nombreux enfants sont atteints de déficiences visuelles. Il leur est difficile de repérer sur l'écran des objets ou des éléments trop petits. De plus, ils ont souvent des problèmes de discernement entre la forme et le fond. C'est pourquoi une attention particulière doit être portée au graphisme. Des graphiques simples, aisément reconnaissables et suffisamment grands sont nécessaires. Il est important de ne pas surcharger les écrans avec des choses inutiles pour ne pas perturber l'enfant et il est conseillé de séparer au maximum les différents éléments. Le graphisme est un élément primordial car il favorise fortement l'attrait du logiciel.

- en ce qui concerne les possibilités auditives et le discernement des couleurs, il n'y a pas de troubles spécifiques aux enfants IMC. Les couleurs peuvent être utilisées dans le but d'aider à dissocier les divers éléments apparaissant sur l'écran, mais une surcharge de ces couleurs peut parfois perturber l'enfant plus que de l'aider dans son travail. Des sons peuvent être associés à certaines actions pour faciliter la compréhension. Les sons et la musique ont un impact important sur l'enfant et rendent le logiciel très attrayant.

- un aspect dynamique doit aussi être envisagé. Certains enfants ont des difficultés à suivre un objet se déplaçant sur l'écran. Par exemple, il leur est parfois difficile de suivre le curseur de la souris lorsqu'ils la déplacent. Il est donc nécessaire de prévoir des déplacements assez lents des objets sur l'écran. Etant donné leurs troubles moteurs, ils maîtrisent rarement leurs appareils d'entrée. C'est pourquoi, il ne faut pas exiger des actions rapides

sur ces appareils. Ils en seraient bien souvent incapables et cela risquerait de provoquer eu eux un sentiment d'impuissance et de frustration, ce qui pourrait aboutir à un désintérêt vis-à-vis du logiciel.

2.2. Niveau intellectuel.

Le quotient intellectuel des enfants IMC est, en général, inférieur à la moyenne. Bien qu'il puisse y avoir des enfants très brillants, la proportion de débiles mentaux dans cette population est relativement élevée. Les enfants avec lesquels nous avons travaillé ne savent généralement ni lire ni écrire, à l'exception peut-être de leur nom, et il leur est difficile de compter. Ils communiquent fréquemment à l'aide de langages symboliques dont un exemple connu sont les symboles BLISS. Il est donc préférable de s'appuyer sur des représentations visuelles des objets et des actions plutôt que sur des représentations textuelles. Ces dernières, lorsqu'elles sont employées, ne doivent contenir qu'un vocabulaire simple, adapté aux enfants de cet âge et toujours être accompagnées d'un dessin.

2.3. Motivations.

Les enfants IMC sont très motivés pour tout ce qui leur apporte des possibilités de communication, d'expression, de jeu ou d'apprentissage. Ils apprécient tout ce qui peut les aider à gagner de l'autonomie. Dans le cadre d'un logiciel, il est important qu'ils perçoivent rapidement les possibilités offertes. Cela les encourage et les motive à acquérir une maîtrise de l'application.

3. La tâche.

Pour éviter des problèmes de compréhension, il est préférable de ne pas présenter des fonctions ou des options trop compliquées. De plus, l'utilisation des appareils d'entrée leur demandant souvent beaucoup de concentration, les enfants IMC se fatiguent très vite face à un ordinateur. Si la tâche à réaliser est trop longue, ils risquent de ne jamais arriver à la

fin. Dès lors, ils se décourageront et se désintéresseront du logiciel. Par contre, si la tâche est constituée de petites séquences d'actions, ils seront satisfaits d'être arrivés à la fin de l'une d'elles et encouragés à en entreprendre une nouvelle.

4. Les règles d'or.

Les règles d'or de la conception sont aussi d'application lorsque l'on développe une interface pour enfants IMC. Cependant, dans ce contexte particulier, certaines d'entre elles revêtent une importance primordiale, alors que d'autres semblent d'un intérêt plus secondaire. Nous allons passer en revue les huit règles décrites par Shneiderman afin d'essayer de dégager leur importance relative.

La possibilité d'employer des *raccourcis* ne paraît pas indispensable. En effet, ceci s'adresse surtout à des personnes expérimentées, ayant une maîtrise du logiciel qui n'est pas à la portée de tous les enfants IMC. De même, donner le *contrôle explicite de l'application* à l'utilisateur n'est peut-être pas toujours essentiel. S'il semble important de laisser la direction des opérations à l'utilisateur dans le cas de programmes de communication ou de programmes ludiques, certains logiciels éducatifs seront plus efficaces s'ils dirigent quelque peu l'enfant IMC.

L'organisation de *séquences d'actions* et la diminution de la *charge de mémoire à court terme* ne sont ni plus ni moins importantes que pour un logiciel classique. Pour diminuer la charge de mémoire à court terme, il serait plus intéressant d'essayer de réduire le nombre d'informations à retenir plutôt que d'augmenter la quantité d'informations présentées à l'écran.

Un soin particulier doit être apporté à *l'uniformité de l'interface* afin de ne pas dérouter l'enfant. Il faut notamment choisir avec soin le vocabulaire et les points de référence employés. Le *feedback informatif* doit également faire l'objet d'attentions particulières, en tenant compte des problèmes de perception des utilisateurs. Dans certains cas, une combinaison son-image pourra être retenue.

Finalement, une *gestion d'erreurs simple* et la possibilité de *revenir en arrière* à tout moment sont d'une importance capitale. En effet, si les enfants IMC savent généralement ce qu'ils veulent faire et comment y parvenir, leurs problèmes moteurs les empêchent souvent de le réaliser avec précision. Il peut par exemple fréquemment leur arriver de vouloir appuyer sur une touche et d'activer celle d'à côté. Le nombre d'erreurs risquant donc d'être important, il faut à tout prix faciliter leur correction et empêcher qu'elles ne mènent à la "catastrophe".

5. Les styles d'interaction.

Certains styles d'interaction fortement appréciés par des utilisateurs valides peuvent devenir un obstacle difficilement surmontable pour des enfants IMC. Il est donc très important de choisir des styles appropriés. Vu le nombre important de handicaps différents, il est souvent utile de prévoir plusieurs types d'interaction afin de permettre l'utilisation du logiciel par des enfants souffrant de troubles divers.

Les *formulaires* sont peu adaptés aux enfants en général, d'une part parce qu'ils manquent d'attraits visuels et d'autre part parce qu'il est nécessaire de savoir lire et écrire pour les utiliser. Pour des IMC, il faut encore ajouter à cela les problèmes posés par l'utilisation du clavier, qui n'est pas toujours un mode d'entrée adéquat.

Les *langages de commande,* en plus de leur complexité, posent les mêmes problèmes de lecture, d'écriture, de manque d'attraits et d'utilisation du clavier que les formulaires. Cependant, un langage de commande relativement réduit fonctionnant avec un système de reconnaissance vocale pourrait être utilisé avec succès par certains enfants IMC. Les *langages naturels* ont les mêmes inconvénients que les langages de commande, mais ils ne peuvent pas encore être utilisés avec des systèmes vocaux et ils ne sont donc généralement pas applicables.

La *manipulation directe,* généralement bien appréciée des personnes valides, peut être accessible à certains enfants IMC, à condition de respecter quelques règles importantes. Les modes d'entrée étant très variés et ayant des caractéristiques différentes tant du point de vue de la

rapidité que de la précision, il est nécessaire d'adapter les objets virtuels à manipuler. Leur taille doit être agrandie et la précision requise pour leur préhension doit être adaptable. Il faut également veiller à la disposition initiale des objets, afin d'en faciliter le repérage et la manipulation, tout en minimisant les déplacements nécessaires pour voyager d'un objet à l'autre. La représentation graphique des différents objets doit également être choisie avec soin et les actions que l'on peut y appliquer doivent être simples à effectuer. Cependant, la manipulation directe ne peut pas être utilisée par tous les enfants IMC: elle n'est par exemple pas accessible aux personnes ne pouvant entrer en communication avec l'ordinateur que via un simple interrupteur.

Les *menus* offrent des caractéristiques très intéressantes. Ils sont simples à utiliser, demandent peu d'apprentissage et sont adaptables à de très nombreux modes d'entrée. Pour être utilisables par un maximum d'enfants IMC, les menus doivent être conçus avec soin. Il faut essayer de minimiser les textes et, si possible, toujours les accompagner d'images simples et explicites. La synthèse vocale peut parfois être employée pour doubler certaines informations visuelles ou confirmer l'item choisi. Le nombre d'options des différents menus doit être relativement limité afin de ne pas surcharger l'écran et de faciliter le repérage des items. Il est également préférable de ne pas choisir une organisation trop compliquée: une organisation linéaire ou une arborescence limitée est préférable à une multi-arborescence ou un réseau de menus. En effet, ces deux premières organisations permettent de savoir plus facilement ou l'on se trouve et sont plus faciles à maîtriser que les deux dernières. Il faut encore pouvoir sélectionner les différents items au moyen de modes d'entrée différents. Il doit être possible d'utiliser la souris et les systèmes pouvant s'y substituer, en évitant de devoir maintenir le bouton de sélection enfoncé pour dérouler les différentes options: cette technique rend problématique l'utilisation de contacteurs pneumatiques. Le clavier doit également permettre la sélection, par exemple par l'utilisation de lettres ou de chiffres représentant les différents items. Dans ce cas, on essayera de regrouper au maximum les touches nécessaires afin de rendre les manipulations plus aisées. Les chiffres sont plus difficiles à mémoriser, mais ils offrent le grand avantage d'être généralement plus facile à trouver sur le clavier. Il est également nécessaire de permettre de choisir

parmis les différentes options au moyen d'un seul interrupteur, en prévoyant un système de balayage automatique des différents items du menu. Moyennant ces différentes adaptations, l'utilisation de menus est le style d'interaction susceptible de satisfaire le plus d'enfants IMC.

6. Les écrans.

Lorsque l'on conçoit un logiciel destiné aux enfants, l'aspect visuel est extrêmement important. Il faut donc soigner le graphisme, utiliser des icônes et des dessins attrayants. Des couleurs gaies peuvent être employées, mais sans excès, afin de susciter l'intérêt sans fatiguer les yeux. Il faut également éviter à tout prix de surcharger l'écran en affichant des informations inutiles ou en présentant trop de commandes à la fois.

Pour faciliter l'utilisation du logiciel par des IMC, il est préférable de bien marquer la différence entre les zones interactives et non-interactives de l'écran. Les zones interactives doivent être regroupées et largement dimensionnées, afin de diminuer les trajets du curseur et de faciliter leur manipulation. Les écrans doivent également être les plus simples possibles et organisés de manière logique. Les informations importantes doivent être suffisamment mises en évidence que pour être remarquées par des personnes ayant certains troubles visuels.

Conclusion.

La conception d'une interface destinée à des enfants IMC demande une étude approfondie. Le plus grand soin doit être apporté à l'étude des possibilités physiques et intellectuelles des utilisateurs afin de prendre les meilleures options possibles. Vu les différents troubles rencontrés, il est essentiel de prévoir une interface adaptable, permettant d'utiliser un maximum de matériels différents. La possibilité de paramétrer l'interface est donc particulièrement importante, afin de pouvoir proposer différents styles d'interaction et différents niveaux d'utilisation.

Chapitre 4 : Le logiciel OrdiThéâtre.

Introduction.

Ce chapitre est exclusivement consacré au logiciel réalisé. Nous exposerons en premier lieu les buts d'OrdiThéâtre et nous le présenterons en détails à travers la méthodologie suivie dans son élaboration. Ensuite, nous l'analyserons selon les critères d'adaptation des logiciels aux IMC décrits au chapitre 3. Cette analyse sera suivie par une brève évaluation d'OrdiThéâtre. Nous justifierons par après le choix du matériel et nous décrirons l'architecture du logiciel ainsi que ses spécifications. Pour terminer, quelques propositions d'améliorations seront présentées.

1. Présentation et buts.

OrdiThéâtre est la simulation d'un jeu de marionnettes sur ordinateur, destiné à des enfants Infirmes Moteurs Cérébraux. Ce logiciel est en premier lieu un jeu et ensuite, un jeu de marionnettes. Nous montrerons donc en quoi le jeu est important pour l'enfant et nous exposerons l'apport que fournissent les marionnettes aux enfants handicapés moteurs. Une brève description d'OrdiThéâtre sera présentée. Une explication plus détaillée sera reprise au point 2. de ce chapitre.

1.1. L'enfant et le jeu.

Dans le développement de l'enfant, le jeu a une importance primordiale. L'enfant a à acquérir un nombre assez incroyable de connaissances, de savoir-faire et de modes de comportement pour devenir un être de plus en plus autonome, responsable et socialisé. Il consacre une grande partie de son énergie à former une relation satisfaisante avec ses parents et les personnes de son entourage. Il se préoccupe aussi de ses besoins physiques. Mais tout le reste de son temps, il le passe dans le jeu,

la découverte et l'apprentissage. Et jouer, c'est apprendre. Les enfants jouent à tous les âges et dans toutes les cultures. L'enfant a besoin du jeu pour grandir, nous le savons. Mais lui, il n'a pas suivi de cours sur l'importance du jeu. Alors pourquoi joue-t-il ? "Le plaisir du jeu est relié à la découverte, à la curiosité qui cherche des réponses. Un jeu perd tout son mystère, sa séduction quand il ne permet plus de rêver, d'inventer, de créer et de recréer. De même, on n'apprend pas réellement quand on n'a pas de motif de le faire. L'enfant joue parce qu'il a du plaisir, l'enfant apprend parce qu'il a du plaisir" [3]. Le plaisir semble donc être l'élément primordial pour lequel l'enfant joue et c'est à travers le jeu que l'enfant non seulement s'éveille, découvre et apprend, mais aussi s'exprime et communique.

Les premiers jouets ne sont que des objets dont la forme, la couleur, le volume, la sonorité attirent l'attention du bébé mais il les explore activement en les touchant des mains, de la bouche, en cherchant à leur faire faire du bruit. Il développe ainsi ses sens et ses connaissances. Au fur et à mesure qu'il grandit, l'enfant assimile ses expériences et se crée un monde intérieur. Il va développer de nouvelles facultés propres aux humains : le langage et la symbolisation. En même temps, l'enfant voit s'enrichir son monde affectif et il s'ouvre de plus en plus à la communication.

J. Piaget, dans sa théorie de l'évolution des jeux, distingue différentes catégories : les jeux d'exercices, les jeux symboliques et les jeux de règles. Dans le premier type, "l'enfant répète pour le seul plaisir d'exercer son activité de la manière la plus complète possible. Après avoir appris à saisir, à balancer, à lancer..., l'enfant répète ces actions pour le seul plaisir d'être cause, pour la joie de dominer ses conduites, de se donner en spectacle sa propre puissance et d'y soumettre l'univers" [4].

Les premiers jeux symboliques apparaissent vers l'âge de deux ans. La transition entre les deux premières catégories s'effectue par le schème symbolique c'est-à-dire que l'enfant reproduit une action, comme faire semblant de manger ou de dormir, en dehors de son contexte et en

[3] Nicole Bolduc [1981], p 88.
[4] C. Vandenplas-Holper [1987], p 46.

l'absence de son objectif habituel. Il exerce d'abord ces conduites sur lui-même et ce n'est que progressivement qu'il les projette sur des objets en faisant dormir son ours ou manger sa poupée. Finalement, il arrive à assimiler un objet à un autre : une boîte d'allumette peut représenter une voiture ou un avion.

A partir de 3-4 ans, les combinaisons symboliques se complexifient et l'enfant représente des scènes de la vie quotidienne. Il s'invente des personnages fictifs qui deviennent des compagnons imaginaires. "La fonction essentielle du jeu symbolique est d'assimiler le réel au Moi. Il donne à l'enfant l'occasion de revivre, en les transposant et en y prenant un rôle actif, des situations pénibles ou désagréables; il permet la réalisation des désirs, la liquidation des conflits" [5].

Entre 4 et 7 ans, apparaît un souci de vraisemblance et d'imitation exacte du réel. Le symbolisme devient collectif, les enfants jouent ensembles.

Entre 7 et 11-12 ans, les enfants passent des jeux symboliques aux jeux de règles qui se jouent nécessairement à plusieurs. L'enfant étend alors son cercle social. Les jeux de règles se prolongent tout au long de la vie sous formes de sports ou de jeux de sociétés.

Le jeu de marionnettes s'inscrit parfaitement dans la catégorie des jeux symboliques. C'est pourquoi il nous semble intéressant de s'attarder sur l'apport de ce type de jeux au développement social, affectif et cognitif de l'enfant. Dans les programmes des écoles maternelles, il est souvent recommandé aux enseignants d'aménager dans la classe différents coins de jeux et de prévoir un matériel varié avec lequel les enfants peuvent se déguiser et jouer différents personnages. Ceci exige que l'enfant analyse son expérience personnelle pour en sélectionner des éléments susceptibles de figurer dans le jeu. Il doit aussi s'adapter aux contraintes de la situation et aux demandes et propositions de ses camarades. Il apprend ainsi à tenir compte d'autrui et dépasse son point de vue égocentrique. En s'efforçant de satisfaire les exigences de son rôle dans le jeu, l'enfant apprend à contrôler son action et à s'imposer une

[5] C. Vandenplas-Holper [1987], p 47.

certaine discipline. Le jeu symbolique contribue donc fortement au contrôle de soi. Par exemple, " un enfant joue le rôle d'un pilote, tombe et a envie de pleurer. Il contrôle sa réaction première puisque "les pilotes ne pleurent pas" " [6]. Pour jouer son rôle, l'enfant part de ses expériences personnelles, de son vécu. Lorsqu'il tient le rôle de la maman, de l'infirmière, de la maîtresse, il se réfère à une personne bien définie. Par le biais de l'expérience de ses partenaires de jeu, il élargit ses propres connaissances. Il observe, par exemple, que des enfants jouant le rôle des parents prennent des attitudes différentes de celles de ses propres parents. Il est alors amené à élargir le concept de père et mère et il devient capable de généraliser ses connaissances. Un autre aspect du jeu symbolique est sa contribution au développement du langage. L'enfant s'exprime en annonçant les différents rôles ("toi, tu es le papa, moi je suis la maman et la poupée est le bébé") et en décrivant les situations ("on joue à la maman qui va au magasin faire ses courses").

La fonction la plus importante du jeu symbolique est qu'il aide l'enfant à s'adapter affectivement et émotionnellement à la société dans laquelle il est appelé à vivre. En effet, il est obligé de s'adapter sans cesse à un monde social d'adultes dont il comprend encore mal les règles. C'est pourquoi il est nécessaire qu'il dispose d'un monde à lui pour faciliter son entrée dans le monde des adultes. " Le jeu permet à l'enfant de surmonter les limitations matérielles qui l'empêchent de se comporter comme il voudrait dans le monde de tous les jours" [7]. Il reproduit ce qu'il observe, ce qu'il vit, ce qu'il comprend, mais à sa façon. Cela lui permet d'affronter la réalité et d'arriver à maîtriser les événements en les rejouant à sa manière. Lorsqu'il a été mis en échec dans la réalité, il lui arrive de rejouer la situation, en l'inversant, et cette fois, en triomphant des difficultés. Il peut, par le jeu, extérioriser ses émotions : ses joies, ses peines, ses angoisses, ses frustrations, ses colères, ses désirs... et ainsi apprendre à les dominer. Cette maîtrise est renforcée par le fait qu'il est actif, qu'il ne subit plus la réalité. Par exemple, une fillette à qui le kinésithérapeute avait placé des attelles en présence de stagiaires et qui avait très mal vécu la situation, a rejoué la scène en devenant le

[6] C. Vandenplas-Holper [1987], p 54.
[7] C. Vandenplas-Holper [1987], p 56.

kinésithérapeute : elle plaçait des attelles à sa poupée tout en expliquant à ses stagiaires. En rejouant une situation où elle s'était sentie passive, totalement dépendante et manipulée, elle devient active et domine la situation. Le théâtre de marionnettes présente un intérêt particulier dans l'optique de maîtrise, d'extériorisation et d'expression des sentiments de l'enfant.

Après avoir décrit l'importance du jeu pour l'enfant, il est utile de s'interroger sur l'influence de l'adulte dans le développement du jeu. Un premier rôle de l'adulte consiste à fournir des jouets qui conviennent à l'enfant à chaque moment de son développement. Dans les jeux symboliques, le choix des jouets est important : des jouets non structurés (papier, bois, sable, matériaux de déguisement) se prêtant à de multiples usages permettent une plus grande créativité que des jouets fortement structurés (répliques miniatures d'appareils ménagers, d'outils). Pour développer l'imagination, il est essentiel que l'enfant ait des interactions avec un adulte auquel il puisse s'identifier et qu'il puisse imiter. Un autre rôle de l'adulte est donc de favoriser le développement du jeu en participant de manière active mais non intrusive. Il doit donc laisser une marge d'intimité suffisante à l'enfant. Ce dernier doit pouvoir rejouer ses expériences sans être observé ni commenté.

1.2. Les marionnettes.

Les marionnettes ne connaissent pas les frontières et les historiens ont bien du mal à certifier leur date de naissance. Leurs formes sont multiples et leurs utilisations nombreuses. Les plus employées sont les marionnettes à gaine et les marottes (manipulées par le bas), les marionnettes à fils (manipulées par le haut), et les marionnettes "vite faites" : figure dessinée sur une main, une cuillère, un morceau de carton, marionnette découpée dans un sac de papier, une chaussette,... En Allemagne, on utilise des marionnettes dans le domaine de l'éducation, de la santé et de la sécurité routière. Dans certains hôpitaux, en France et aux USA, la marionnette est mise au service de la psychiatrie, et l'on s'en sert aussi pour l'accueil des enfants malades ou blessés : elle apaise les peurs et les angoisses. En Tchécoslovaquie, des cours très poussés sont

organisés systématiquement pour les instituteurs. La marionnette occupe une place très importante dans l'éducation des enfants de l'Est. En Belgique, à part quelques spectacles pour enfants et quelques initiatives en thérapie, les marionnettes restent insuffisamment employées.

Pourtant, le théâtre de marionnettes remplit parfaitement la fonction la plus importante du jeu symbolique qui consiste à aider l'enfant à s'adapter affectivement et socialement à son entourage. L'enfant, au travers des marionnettes, peut apprendre à maîtriser, à extérioriser et surtout à exprimer ses sentiments. Tout en suscitant l'imagination et la créativité, ce jeu lui offre des marionnettes dans lesquelles il peut se projeter, ainsi que son entourage et son environnement. Par cette projection, qui est grandement facilitée par le pouvoir d'attrait des marionnettes sur l'enfant, il peut placer une distance entre lui et la réalité, les événements, les émotions, puisque ce n'est plus lui-même mais la marionnette qui subit la situation. Par la mise en scène, il exprime certains sentiments de manière plus libre, plus naturelle, plus concrète que s'il devait les expliciter verbalement, le jeu étant le moyen naturel d'expression de l'enfant. Un observateur attentif, psychologue ou autre, peut alors l'"écouter" et ainsi recevoir ce que l'enfant a à dire. Un autre aspect intéressant des marionnettes est qu'elles permettent à l'enfant de se dépasser. En effet, il peut leur faire effectuer des actions qui lui sont, soit impossibles -par exemple, une marionnette qu'il fait tournoyer dans les airs le transformera en un fabuleux acrobate-, soit interdites -plutôt que de frapper un autre enfant, il déchargera son agressivité en faisant se disputer les marionnettes. Celles-ci ont un pouvoir infaillible pour récupérer les sautes d'humeur et l'agressivité et les mettre au service d'une communication la plus parfaite possible, nécessaire à la compréhension. Les enfants, tout en rejouant, créent l'illusion de la vie. Les situations dans lesquelles ils vont mettre les marionnettes peuvent être effrayantes et donc satisfaire leur agressivité sans qu'ils aient peur. Même si certaines situations sont menaçantes, les enfants savent en fait qu'ils "font semblant".

L'importance qu'il faut attribuer à la marionnette est bien évidente. A elle seule, elle incite les enfants à exercer leur créativité, leur sensibilité, leur savoir-être et leur savoir-faire dans bien des domaines.

Aux enfants infirmes moteurs cérébraux, elle offre de merveilleuses possibilités d'expression et de communication.

Les marionnettes, vues sous l'optique de moyen d'expression, sont particulièrement importantes pour les enfants IMC qui n'ont pas l'usage de la parole. Il s'agit d'enfants que des troubles mécaniques privent de toute communication orale. Ils ne peuvent s'exprimer et se font comprendre très difficilement. Or parler, c'est mettre à distance ce que l'on ressent. La parole offre une possibilité de contrôle dont ces enfants sont privés. Ils disposent pourtant d'un langage intérieur mais celui-ci est difficilement accessible et compréhensible et de plus, toute parole a besoin d'être entendue. Un exemple cité par Mme de Barbot, psychologue, illustre parfaitement cette idée : " il s'agit d'une fillette qui se sert d'un communicator Canon, sur lequel elle peut taper des messages. Il y a un an, elle l'utilisait peu. Un jour, à l'heure du repas, je l'ai trouvée en pleine "crise de nerfs" : tout le monde avait compris pourquoi (un autre enfant lui avait pris une partie de son dessert) et on trouvait donc inutile de le lui faire dire. J'ai insisté néanmoins pour qu'elle l'écrive grâce à son communicator... et elle s'est calmée, dès que sa crise émotionnelle a été "mise en mots". Depuis qu'elle se sert plus habituellement de sa Canon, cette enfant est plus calme." Ces enfants IMC ont donc peu l'occasion de parler. C'est pourquoi il faut réfléchir aux moyens de donner à un enfant très handicapé la possibilité d'extérioriser ce qu'il ressent, ce qu'il éprouve, que ce soit violent, angoissant, gai ou drôle. Mais ce n'est pas toujours facile quant on se trouve face à des enfants qui ne peuvent ni verbaliser, ni manipuler. On ne peut pas indéfiniment les laisser regarder jouer les autres en supposant qu'ils participent suffisamment à ce qui est représenté pour en tirer quelques bénéfices.

Les marionnettes fournissent un bon moyen d'expression pour les enfants capables de les manipuler. Même si ces enfants ne parlent pas, ils s'expriment et communiquent à travers la marionnette. Un moniteur attentif peut alors les comprendre et percevoir ce qu'ils ont à dire. Mais qu'en est-il des enfants incapables de manipuler les marionnettes ? Dans ce cas, une simulation sur ordinateur semble constituer une aide non négligeable.

1.3. Le logiciel Ordithéâtre.

Ordithéâtre a été réalisé sous la direction de Mme Noirhomme (FNDP, institut d'informatique) et de Mr Mercier (FNDP, département de psychologie), en collaboration avec le service IMC du centre hospitalier universitaire de Kremlin-Bicêtre. Celui-ci accueille des enfants IMC âgés de 6 à 12 ans dont les capacités intellectuelles leur permettent de suivre une scolarité primaire.

Le logiciel OrdiThéâtre a pour but d'offrir une possibilité de communication aux enfants qui n'ont pas les capacités motrices pour utiliser un véritable théâtre de marionnettes. C'est un moyen de donner la parole à des enfants qui en sont souvent privés. Il leur permet d'exprimer leurs sentiments, d'énoncer leurs problèmes éventuels. Ceci est particulièrement important pour des enfants qui, tous les jours, doivent faire face à leur handicap. Beaucoup d'IMC ont bien du mal à l'accepter et les laisser se replier sur eux-mêmes n'est certainement pas une bonne solution. Mais s'exprimer verbalement, pour un enfant, n'est pas toujours aisé surtout lorsqu'il s'agit de sujets difficiles à aborder. Les marionnettes fournissent un merveilleux moyen de communication. C'est pourquoi, pour des enfants incapables de les manipuler, nous avons créé des "marionnettes informatiques" au travers desquelles ils parlent de choses tristes, gaies, angoissantes ou drôles; à l'aide desquelles ils racontent, inventent, imaginent et créent. OrdiThéâtre peut ainsi servir de base à une thérapie. En effet, un psychologue peut observer l'enfant et l'écouter parler de ses problèmes. Il est possible aussi d'en dégager une utilisation pédagogique : l'animation des marionnettes fait appel à des notions telles que aller en haut, en bas, à droite, à gauche, se tourner, s'asseoir, se coucher, se mettre debout, écouter... Enfin, OrdiThéâtre peut aussi être utilisé dans un but uniquement ludique à la grande joie des enfants.

Le logiciel a été conçu de manière à donner à l'enfant une complète autonomie dans son utilisation. Pour cela, le moniteur adapte un certain nombre de paramètres en fonction du handicap de l'enfant. Ces paramètres définis, l'enfant peut jouer seul sans risque de commandes dangereuses qui l'empêcheraient de continuer. Cette indépendance est importante pour des enfants qui sont constamment portés, manipulés,

aidés dans la moindre activité et qui dépendent totalement de la présence d'un adulte à leur côté. Le fait d'être autonome dans l'utilisation d'OrdiThéâtre, de décider et de concrétiser eux-mêmes leurs choix sans l'aide d'une tierce personne, leur procure une grande joie et c'est là déjà un merveilleux apport du logiciel.

OrdiThéâtre comprend trois parties. La première est destinée au moniteur afin de préparer le jeu en fonction des capacités de l'enfant. La deuxième consiste soit en la visualisation d'une scène créée auparavant et enregistrée, soit en la sélection d'un décor, de marionnettes et d'un thème musical en vue de créer une nouvelle scène. Le jeu, c'est-à-dire la création de la scène, se déroule dans la troisième partie : l'enfant anime les marionnettes choisies précédemment, dans le décor et au son de la musique sélectionnés. Il a la possibilité d'enregistrer son animation s'il désire la conserver pour la revoir par la suite.

1.3.1. Première partie : préparation par le moniteur.

Le moniteur doit déterminer un certain nombre de paramètres afin de préparer le logiciel en fonction de l'enfant. En effet, vu la diversité des troubles des enfants IMC, il est nécessaire de prévoir différentes possibilités d'utilisation du logiciel. Ces paramètres peuvent être enregistrés en fonction du nom de l'enfant afin de ne pas devoir les redéfinir à chaque utilisation. Cependant, il est toujours possible de les modifier en cours de jeu, ce qui augmente la souplesse de l'application.

Afin d'être accessible à un maximum d'enfants, l'activation des commandes du logiciel se fait par le positionnement d'un curseur dans une ligne de commandes située dans le bas de l'écran. Cette ligne regroupe des commandes illustrées par un dessin simple et explicite. Le déplacement du curseur et la validation seront fonction du mode d'utilisation choisi, à savoir :

* la souris : déplacement du curseur à l'aide de la souris et validation en cliquant. Ce mode permet également l'utilisation d'un joystick, d'une track-ball, d'un casque ultrasonique ou de tout autre matériel s'adaptant à la sortie souris.

* le clavier-flèches : déplacement du curseur dans la ligne de commandes à l'aide des flèches <-, -> ou des lettres K et L, et validation au moyen de <RETURN>. Le clavier normal peut être remplacé par un mini-clavier, un clavier étendu ou un jeu d'interrupteurs raccordés à la Magic-box.

* le clavier-défilement : déplacement automatique du curseur dans la ligne de commandes et validation au moyen de <RETURN>. La validation peut également s'effectuer par le biais d'un interrupteur si l'on utilise la Magic-box.

A l'aide de menus déroulants, le moniteur détermine le mode d'utilisation le mieux adapté à l'enfant. Ceci constitue le paramètre le plus important car c'est celui qui permet un accès aisé à l'ordinateur.

Un second paramètre est la possibilité pour le moniteur d'enregistrer une trace pendant les différentes phases de jeu. La trace consiste en l'enregistrement de tous les choix et de toutes les manipulations de l'enfant. Ceci peut être intéressant en cas d'évaluation de l'enfant et de ses progrès.

Le moniteur peut également décider des commandes qui seront proposées à l'enfant lors de la phase d'animation des marionnettes. Pour créer une scène, l'enfant sélectionne des commandes affichées dans le bas de l'écran et la marionnette les effectue au fur et à mesure de leur sélection. Cependant, il se peut que lors des premières utilisations l'enfant soit perdu face à un nombre trop élevé de commandes proposées. Le moniteur peut dès lors ne lui en présenter que quelques unes et augmenter leur nombre petit à petit.

Cette brève présentation de la première partie du logiciel sera détaillée dans le point 2 de ce chapitre.

1.3.2. Deuxième partie : visualisation d'une scène créée ou préparation d'une nouvelle scène.

Dans cette partie ainsi que dans la troisième, l'enfant est autonome. Il décide et introduit ses choix via son mode d'accès à l'ordinateur sans l'intervention du moniteur. Au départ, il peut soit aller regarder une scène qu'il a créée auparavant, soit aller sélectionner les éléments nécessaires à la création d'une nouvelle scène.

S'il choisit de regarder une scène, l'enfant doit sélectionner celle qu'il désire revoir. Les scènes qu'il a enregistrées lui sont présentées une à une et il peut ainsi les passer en revue avant de faire son choix. La présentation d'une scène consiste en l'affichage des marionnettes choisies, sur le décor, avec le titre éventuel que l'enfant lui aurait donné lors de la création de la scène. A ce moment, l'enfant a la possibilité de jeter une scène qu'il n'a plus envie de conserver. C'est pour éviter la suppression de scènes d'autres enfants qu'il n'a accès uniquement qu'à celles qu'il a créées. Lorsqu'il sélectionne une scène, celle-ci se rejoue au son du thème musical choisi. Il peut alors la modifier, la continuer ou la laisser telle quelle et aller en regarder une autre.

S'il choisit l'option de créer une nouvelle scène, l'enfant doit sélectionner son décor, ses marionnettes et un thème musical, celui-ci n'étant pas obligatoire. Il choisit ces éléments dans l'ordre qu'il veut. Il peut sélectionner un maximum de quatre personnages et prendre plusieurs fois le même s'il le désire. Lorsqu'il a effectué ces choix, il passe à la phase suivante qui est l'animation des marionnettes.

1.3.3. Troisième partie : création d'une scène.

Le décor sélectionné apparaît sur l'écran et le thème musical se joue. L'enfant doit alors faire entrer ses marionnettes en scène et créer son histoire. Il anime les personnages en sélectionnant des commandes situées dans le bas de l'écran. Chaque sélection entraine l'exécution immédiate de la commande par la marionnette concernée. L'enfant

dispose d'une caméra qu'il peut actionner ou couper à tout moment pour enregistrer ce qu'il veut conserver.

Ceci conclu la présentation brève du logiciel et la description précise de ses buts. Une explication détaillée d'OrdiThéâtre est proposée ensuite à travers la méthodologie suivie dans son élaboration.

2. Methodologie.

Nous exposons dans cette section la méthodologie suivie dans l'élaboration du logiciel, de la rédaction du cahier des charges en septembre 1989 à la réalisation de la version finale de l'application en juillet 1990. Les choix et les modifications effectuées suite aux divers problèmes rencontrés tout au long de l'année sont expliqués et justifiés. OrdiThéâtre, ainsi que son évolution durant les 10 mois de sa réalisation, sont ainsi présentés en détails.

2.1. Le cahier des charges avant le stage.

Ce mémoire a débuté par l'élaboration du cahier des charges. Celui-ci a été conçu en plusieurs étapes, de septembre 1989 à fin octobre. Le logiciel ayant été demandé par Mr Mercier du département de psychologie, il nous a exposé lors d'une entrevue début septembre les grandes fonctionnalités et les exigences de ce logiciel. Celles-ci étaient principalement : les choix par l'enfant IMC d'un décor, de marionnettes et si possible d'un thème musical; l'animation de ces marionnettes dans le décor et au son de la musique; et la possibilité d'enregistrer cette animation et la revoir par après. Nous avons, à partir de ces indications, rédigé une première ébauche du cahier des charges répondant à ces exigences mais nous n'avions jamais eu de contact avec la population concernée. Nous ne connaissions donc pas précisément les différents problèmes et les diverses capacités des enfants IMC. Cette première version était donc sujette à de nombreuses corrections et précisions à faire lors de notre stage à Paris au service IMC de l'hôpital de Kremlin-

Bicêtre. En fait, cette première version du cahier des charges comprenait les idées de base du logiciel, à affiner sur le lieu de stage suivant les observations retirées de notre contact avec les enfants. Voici cette première version :

"Ce logiciel concerne la simulation d'un jeu de marionnettes et est destiné aux enfants n'ayant pas les capacités motrices nécessaires à l'utilisation d'un véritable théâtre. Il s'adresse en premier lieu aux enfants IMC mais il pourra être étendu aux handicapés mentaux.

En conséquence, l'interface de ce logiciel devra être bien adaptée aux potentialités de cette population. Celles-ci seront étudiées précisément lors du stage à Paris. Nous pouvons cependant déjà partir sur les bases suivantes :

- l'utilisation d'une seule touche au clavier pour les IMC les plus profonds.
- l'utilisation de la souris pour les enfants capables de la manipuler.

La sélection des fonctions du logiciel se fera par un système de balayage dans des menus permettant l'utilisation d'une seule touche, ou par l'utilisation directe de la souris dans le menu. Ces menus devront tenir compte des problèmes visuels et autres (problèmes de lecture, de mémorisation...) en étant constitués de dessins suffisamment grands et explicites.

Le logiciel comprendra trois parties : la première sera destinée au moniteur afin de préparer l'utilisation du logiciel par l'enfant en fonction de ses capacités; la deuxième partie consistera en une phase de sélection d'un décor, d'un certain nombre de personnages et d'un thème musical, par l'enfant lui-même; la dernière partie sera le jeu proprement dit : animation des différentes marionnettes et enregistrement.

PARTIE 1.

- choix par le moniteur du mode d'utilisation (souris ou une seule touche)
- choix du mode de jeu :
    - visualisation d'une scène déjà enregistrée
    - création d'une nouvelle scène
- dans le cas de la création d'une nouvelle scène, choix du type d'enregistrement :
    - enregistrement de toutes les manipulations, durant la phase de sélection et durant la phase de jeu
    - enregistrement de la phase de jeu uniquement
    - enregistrement de la phase de sélection uniquement
    - pas d'enregistrement.

PARTIE 2.

1. choix du thème musical.
    - proposition de 6 thèmes illustrés par un dessin simple
    - lorsque l'enfant a sélectionné un thème, celui-ci se joue et la confirmation du choix est demandée.

Lorsque le thème est confirmé, il se joue jusqu'à la fin du jeu (donc, y compris pendant la sélection du décor et des personnages).

2. choix du décor.
    - présentation de plusieurs pages comprenant 4 décors + 1 case pour le passage à la page suivante (la page suivant la dernière étant la première).

Lorsque l'enfant choisit un décor, celui-ci s'affiche sur l'écran en taille réelle et l'enfant confirme ou pas son choix.

3. choix des personnages.
    - présentation sur les 2/3 supérieurs de l'écran d'un certain nombre de personnages + une case "passage page suivante" + une case "fin".

- lorsque l'enfant sélectionne un personnage, celui-ci s'affiche dans le 1/3 inférieur de l'écran, à côté des personnages déjà sélectionnés (sélection de 4 personnages maximum).
- lorsque la sélection est finie, le décor s'affiche dans la zone supérieure avec les personnages choisis. L'enfant confirme ou pas son choix. Dans la négative, il choisit le personnage à remplacer et resélectionne un nouveau personnage.

Lorsque le choix est confirmé, le rideau se ferme et se rouvre pour la phase suivante.

PARTIE 3.

Le décor apparait sur l'écran dans les 2/3 supérieurs et les personnages apparaissent dans le 1/3 inférieur ainsi qu'une case de fin de jeu. L'enfant sélectionne le personnage qu'il veut animer. La ligne de commandes de ce personnage remplace alors le tiers inférieur de l'écran. Le personnage se trouve dans la première case. Les cases suivantes (leur nombre sera discuté en fonction de la taille de l'écran) comprennent des commandes relatives à ce personnage + 1 case de passage à la page suivante des commandes. Les commandes possibles sont illustrées par un dessin simple.
Ces commandes sont :
- entrer/sortir de scène
- déplacement d'une distance fixe vers la gauche, vers la droite (éventuellement haut, bas ou saut)
- position assise, debout, couchée
- parler (agitation de la bouche et/ou des bras)

La sélection de la première case (le personnage) entraine le retour à la ligne comprenant les 4 personnages.

Ceci est une première approche du cahier des charges. Celui-ci sera défini plus précisément durant le stage."

Un certain nombre de remarques concernant ce premier cahier des charges nous ont été fournies, avant notre départ, par Madame Wanet, assistante au département de psychologie.

En premier lieu, le choix du mode de jeu dans la partie 1 doit être proposé à l'enfant et non au moniteur. C'est l'enfant qui doit décider s'il veut visualiser une scène qu'il avait créée ou en réaliser une nouvelle. De même pour l'enregistrement de la scène, l'enfant doit avoir la possibilité de le brancher ou de le couper à tout moment de l'animation.

En ce qui concerne la deuxième partie, l'ordre des sélections de la musique, du décor et des marionnettes ne doit pas être imposé. L'enfant a le droit de choisir ces éléments dans l'ordre qu'il désire. Pour le thème musical, il est intéressant qu'il ait un échantillon de la musique au moment de la sélection, mais celle-ci ne doit se jouer que pendant l'animation pour éviter des risques de lassitude.

Enfin, l'enfant doit pouvoir, à tout moment de la sélection, revenir en arrière pour modifier ses choix. Il est nécessaire de lui fournir la possibilité de passer d'un menu à l'autre (décor, marionnettes, musique) au cas où il changerait d'avis en ce qui concerne les sélections qu'il a déjà réalisées.

Ces premières modifications effectuées, nous sommes partis sur le lieu du stage pour observer et comprendre les difficultés et les capacités des enfants Infirmes Moteurs cérébraux et réaliser une version précise et définitive du cahier des charges.

2.2. Le stage et le cahier des charges définitif.

Nous nous sommes donc rendus à Paris pour un stage de 6 semaines au service IMC du centre hospitalier universitaire de Kremlin-Bicêtre. Ce service, dirigé par Madame Truscelli, Docteur en médecine, intègre une école spécialisée accueillant une trentaine d'enfants, âgés de 6 à 12 ans, tous ayant les capacités intellectuelles nécessaires à l'achèvement d'une scolarité primaire. Leur encadrement y est assuré par des institutrices, des infirmiers(ères), des ortophonistes, des ergothérapeutes, des

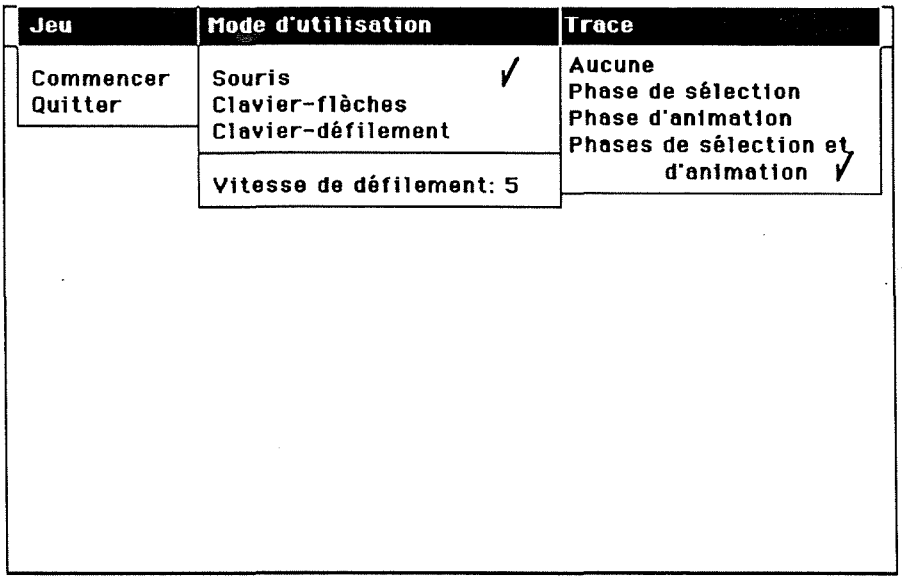
kinésithérapeutes, des psychologues et des médecins. Durant notre séjour, nous avons observé les enfants, aussi bien dans les classes que pendant les séances de kinésithérapie, la récréation et les repas, mais notre attention s'est principalement portée sur les séances de travail avec l'ordinateur. Le personnel étant fortement disponible, nous avons pu obtenir auprès de lui tous les renseignements dont nous avions besoin. Avec la collaboration de Mme C. Charrière, ergothérapeute, et Mme de Barbot, psychologue, nous avons élaboré un scénario précis de notre logiciel.

Dans ce service IMC, l'ordinateur est surtout utilisé dans un but pédagogique (apprentissage de la lecture par exemple). Chaque ordinateur est adaptable à chaque enfant en fonction de son handicap particulier, grâce à l'utilisation de nombreuses cartes d'extension et matériels spécialisés. Ainsi, Olivier utilise un mini-clavier; Mathilde, un clavier normal muni d'un cache clavier; Olivier (un autre) emploie une licorne; Cécile travaille à l'aide d'un casque ultrasonique; Perrine n'utilise qu'un seul interrupteur... Pour les enfants sachant utiliser le clavier (ou un matériel équivalent), l'utilisation d'une seule touche avec un système de défilement d'un curseur dans les menus est trop lent et vite lassant. C'est pourquoi nous avons ajouté à notre cahier des charges, une possibilité d'utilisation supplémentaire : le clavier-flèches. L'enfant déplace lui-même le curseur à l'aide de 2 touches du clavier et valide sa sélection avec une troisième. Les enfants qui utilisent le défilement ne travaillant pas tous avec la même rapidité, nous avons pris la décision de rendre la vitesse de défilement réglable par le moniteur. Nous avons aussi défini plus précisément la notion de "menu" dans lequel se déplace le curseur. Il s'agit en fait de lignes de commandes situées dans le bas de l'écran. Pour favoriser l'uniformité du logiciel et sa compréhension, toute commande se fera par le positionnement du curseur dans cette ligne de commandes et ce, quel que soit l'endroit où l'on se trouve dans le logiciel. La découpe d'OrdiThéâtre en 3 parties a été maintenue et celles-ci ont été précisées en tenant compte de nos observations et des remarques pertinentes de Mesdames Charrière et de Barbot. La description complète de ces 3 parties présentée ci-après, constitue, ajoutée aux remarques et modifications énoncées ci-dessus, la version définitive du cahier des charges.

Partie 1.

Pour ne pas complexifier la tâche de l'enfant, cette partie est réservée au moniteur. Les divers paramètres se trouvent dans des menus déroulants, le moniteur y accédant avec la souris. La barre de menus étant cachée pendant le jeu, elle est accessible à tout moment au moyen d'une touche de contrôle.

Ecran 1: cahier des charges.



La commande d'arrêt de jeu <Quitter> est réservée au moniteur pour éviter que des enfants ne quittent le jeu accidentellement et, ensuite, ne sachent plus y revenir. C'est également le moniteur qui lance le jeu une fois les paramètres définis.

La trace est l'enregistrement de tous les choix et de toutes les manipulations de l'enfant. Elle peut soit ne pas être actionnée, soit être actionnée durant la phase de sélection (partie 2), durant la phase d'animation (partie 3) ou durant ces deux phases. Etant enregistrée dans un fichier séparé, celui-ci peut-être facilement imprimé. Voici un exemple de trace effectuée pendant la phase d'animation.

--- OrdiThéâtre : Fichier trace ---

Utilisateur : XAVIER
Date : 28/8/1990 --- Heure : 10:54:34

Paramètres utilisés :
Mode d'utilisation : souris
Temporisation du curseur (défilement) : 1,50 sec.
Organisation des lignes de commandes :
L1 : changer de personnage / changer ligne de commandes / caméra / entrer-sortir / gauche / droite / haut / bas /
L2 : changer de personnage / changer ligne de commandes / lever / asseoir / coucher / agiter / pivoter à gauche / pivoter à droite /
L3 : changer de personnage / changer ligne de commandes / musique / stop/
L4 :
Type de trace : trace pendant la sélection et l'animation
Enregistrement automatique : non

0m1s44c : Appuie sur "m" -> <commande>-<m>
0m6s54c : Clique en (322 , 315) -> créer une scène
0m11s81c : Clique en (39 , 229) ->
0m13s11c : Clique en (41 , 317) -> décors
0m17s98c : Clique en (229 , 308) -> suivant
0m22s1c : Clique en (357 , 314) -> O.K.
0m27s28c : Appuie sur "g" ->
0m30s94c : Clique en (144 , 305) -> personnages
0m42s54c : Clique en (41 , 298) -> précédent
0m50s21c : Clique en (246 , 302) -> prendre
1m9s91c : Clique en (374 , 317) -> modifier
1m15s98c : Clique en (144 , 313) -> effacer personnage 1
1m17s1c : Clique en (81 , 310) -> O.K.
1m18s81c : Clique en (81 , 309) -> précédent
1m21s81c : Clique en (50 , 144) -> clique sur un personnage
1m22s81c : Clique en (258 , 299) -> prendre
1m29s68c : Clique en (422 , 322) -> O.K.
1m37s58c : Clique en (390 , 307) -> créer
1m40s51c : Clique en (228 , 232) -> O.K.
1m47s24c : Clique en (184 , 303) -> caméra

1m48s91c : Clique en (237 , 311) -> entrer-sortir
1m50s64c : Clique en (274 , 308) -> gauche
1m51s94c : Clique en (231 , 260) -> clique sur un personnage
1m55s81c : Clique en (230 , 336) -> lâcher le personnage
2m0s94c : Clique en (336 , 309) -> droite
2m2s31c : Clique en (76 , 314) -> changer ligne de commandes
2m3s68c : Clique en (347 , 310) -> agiter
2m5s51c : Clique en (226 , 312) -> asseoir
2m6s58c : Clique en (425 , 316) -> pivoter à gauche
2m7s88c : Clique en (108 , 294) -> changer ligne de commandes
2m8s88c : Clique en (395 , 306) -> stop
2m11s38c : Clique en (325 , 231) -> annuler
2m12s21c : Clique en (415 , 304) -> stop
2m13s14c : Clique en (223 , 229) -> O.K.
2m40s88c : Clique en (465 , 315) -> stop
2m42s14c : Clique en (152 , 232) -> O.K.
2m45s54c : Appuie sur "m" -> <commande>-<m>
2m47s64c : Clique en (44 , 3) -> menu: quitter

Il est intéressant de conserver une trace lorsque l'on veut évaluer les progrès de l'enfant d'une séance à l'autre. Les erreurs de manipulation (par exemple, cliquer en dehors de la ligne de commandes) ainsi que le temps écoulé entre 2 actions sucessives peuvent donner des indications sur le niveau de compréhension atteint par l'enfant vis-à-vis du logiciel. Une évaluation plus psychologique est également possible à travers des éléments tels que les marionnettes, le décor et le thème musical choisis par l'enfant lors de différentes séances. Il est intéressant de savoir si l'enfant prend toujours le premier élément qui lui est présenté ou s'il les passe en revue avant de se décider. Joue-t-il toujours avec les mêmes marionnettes d'une fois à l'autre ou change-t-il fréquemment de contexte de jeu? Revient-il souvent en arrière pour modifier ses sélections ou sait-il ce qu'il veut dès le départ? Branche-t-il la musique ou préfère-t-il jouer en silence? Toutes les réponses à ces questions peuvent constituer un apport précieux au psychologue travaillant avec l'enfant.

Une fois les paramètres définis, le moniteur lance le jeu au moyen de la commande <commencer> du menu "jeu" et l'on entre directement dans la partie 2 dans laquelle l'enfant est autonome.

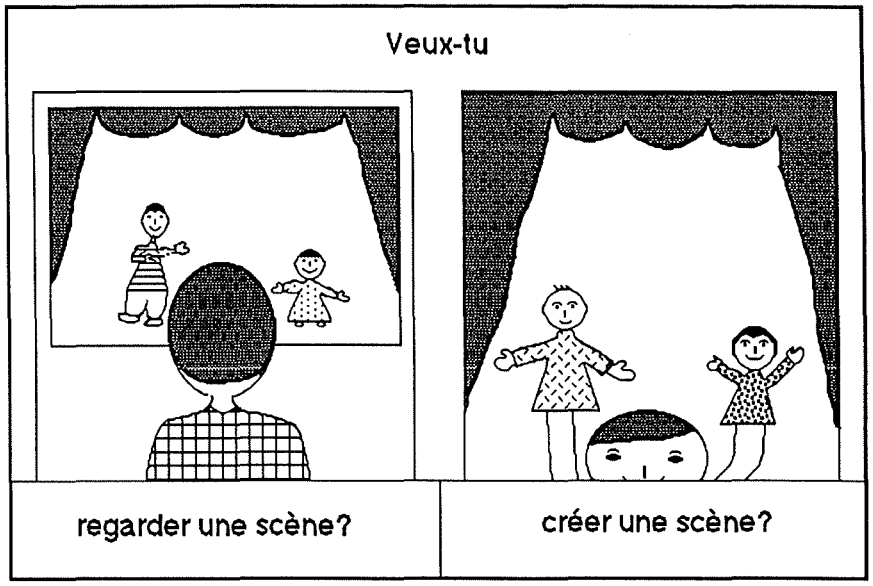
Partie 2.

Le choix entre les deux grandes options du logiciel "regarder une scène" et "créer une scène" est proposé à l'enfant. Celui-ci indique son choix en plaçant le curseur dans la ligne de commandes du bas de l'écran.

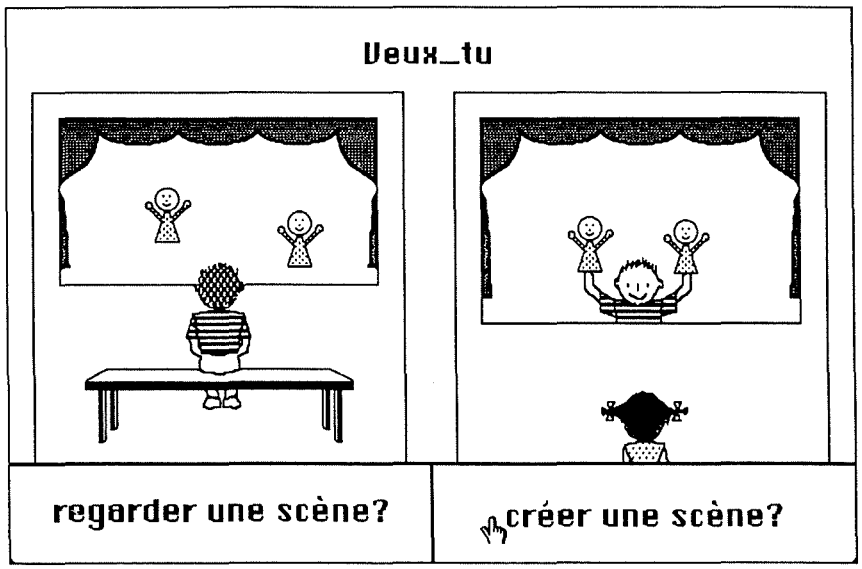
Les dessins intervenant fortement dans l'attrait de ce type de logiciel vis-à-vis des enfants, il est primordial de présenter un graphisme de qualité. Vu le nombre de dessins à réaliser (19 dessins pour une seule marionnette!), nous avons demandé la collaboration de Melle A. de Theux, graphiste. Ce travail aurait exigé, de notre part, énormément de temps pour un résultat de qualité nettement inférieure.

Pour chaque écran présenté par la suite, nous montrons en premier lieu l'idée d'écran telle que nous l'avions établie et dessinée dans le cahier des charges et ensuite, la réalisation par la graphiste de cet écran tel qu'il apparaît dans la version finale d'OrdiThéâtre. Les différences éventuelles entre les deux versions des écrans sont le résultat de changements survenus durant la période de conception du logiciel. Les explications de ces modifications seront détaillées après la présentation complète du cahier des charges.

Ecran 2 : cahier des charges.



Ecran 2 : version finale.



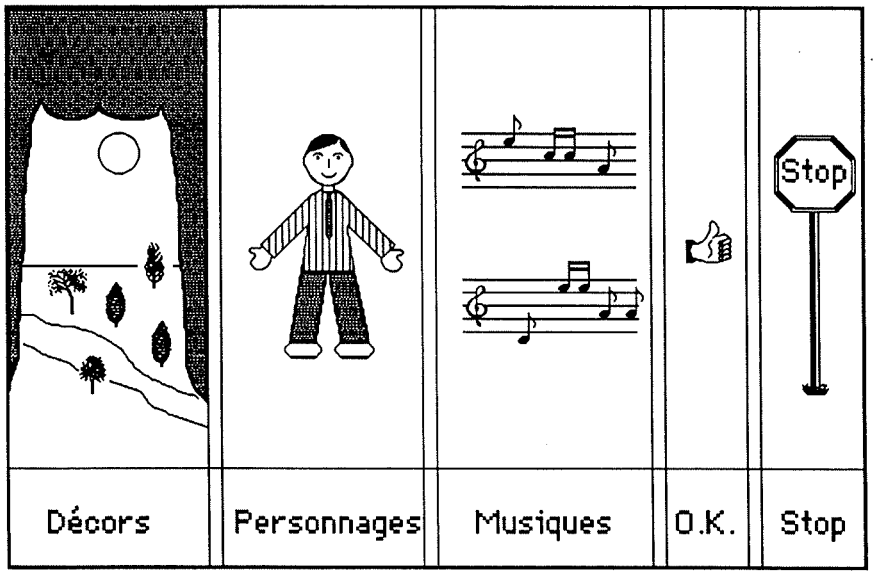
En ce qui concerne le style de dessin à réaliser pour les décors et les marionnettes, il nous a été conseillé de ne pas trop charger les décors et de plutôt fournir des personnages détaillés mais suffisamment vagues pour permettre plusieurs interprétations. En ce qui concerne les commandes, il nous a été recommandé de les illustrer par des sigles

familiers aux enfants (par exemple, les sigles routiers) et facilement reconnaissables.

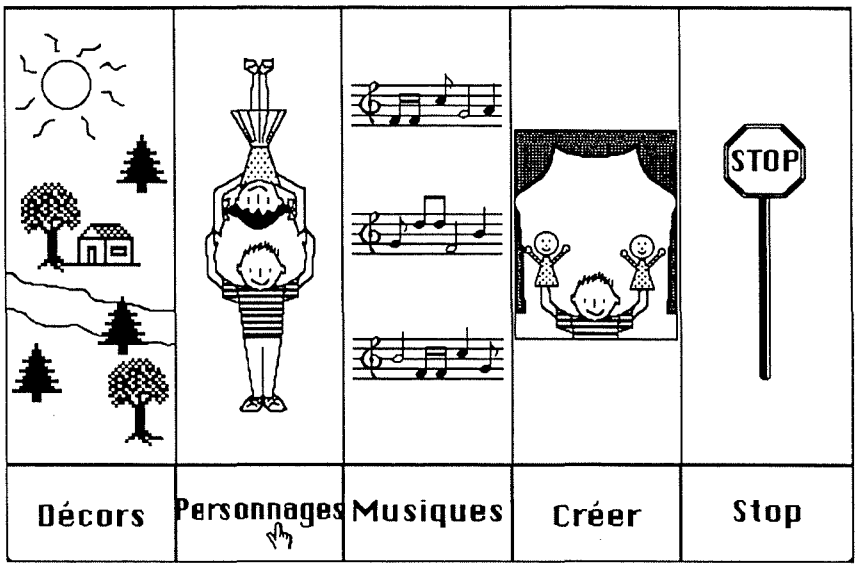
* Option "créer une scène".

En choisissant cette option, l'enfant passe à la sélection du décor, des marionnettes et de la musique. Aucun ordre ne lui est imposé dans ses choix et il peut toujours revenir en arrière pour les modifier au cas où il changerait d'avis. La sélection de l'option voulue se fait par le positionnement du curseur dans la ligne de commandes et la validation de la commande selon le mode d'utilisation.

Ecran 3 : cahier des charges.

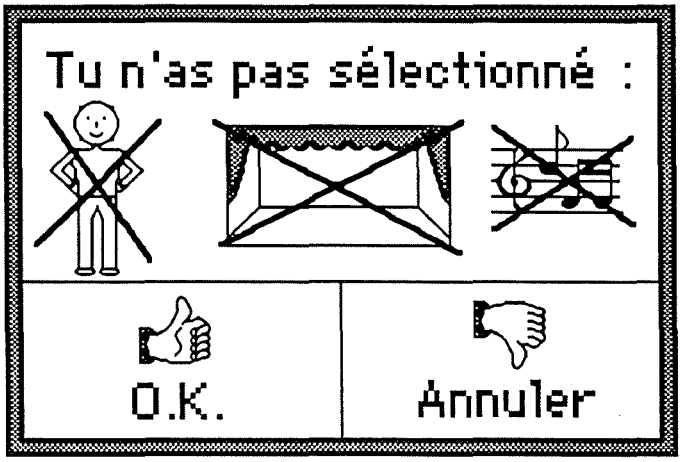


Ecran 3 : version finale.

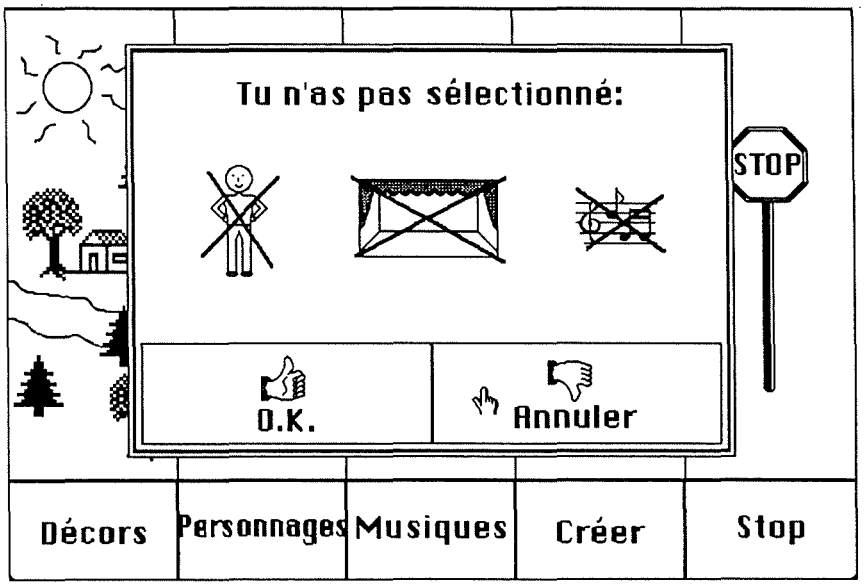


Le cahier des charges prévoit, après la sélection de l'option <décors>, <personnages> ou <musiques>, le passage direct aux écrans de choix correspondants. Il en est de même pour la commande <stop> qui implique le retour immédiat à l'écran de présentation des 2 options principales d'OrdiThéâtre (écran 2). Par contre, la commande <ok> entraine l'affichage d'une fenêtre d'avertissement dans le cas où l'enfant a oublié de sélectionner certains éléments. L'enfant peut alors revenir à l'écran de choix du décor, des personnages et de la musique ou passer, comme il l'avait demandé, à l'écran d'animation. Il se peut, en fait, que l'enfant veuille intentionnellement jouer sans musique ou sans décor.

Fenêtre 1 : cahier des charges.



Fenêtre 1 : version finale.

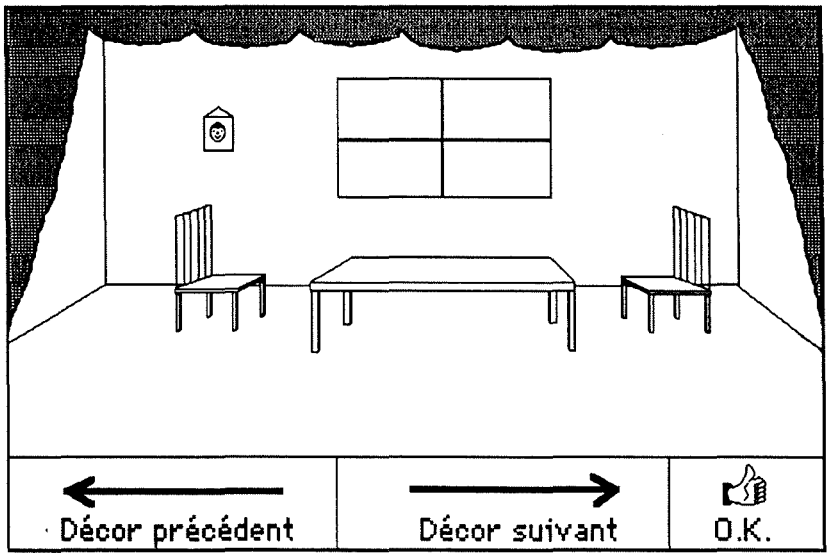


1. choix du décor.

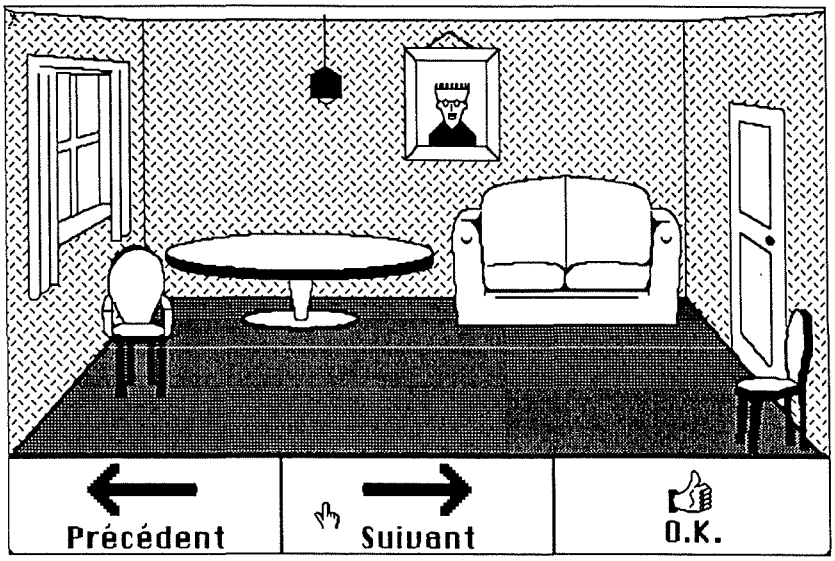
Après la sélection de la commande <décor> de l'écran précédent, apparaissent un décor et les commandes <décor précédent>, <décor suivant> et <ok>. Les deux premières permettent de passer en revue les différents décors disponibles (le déroulement est circulaire), et la troisième implique la sélection du décor présenté sur l'écran et le retour à

l'écran précédent. Dans la première version du cahier des charges, rédigée avant le stage, nous avions prévu de présenter 4 décors par page. Nous nous sommes rendus compte à Paris que de nombreux enfants IMC souffraient de déficiences visuelles. Il est donc préférable de ne pas leur présenter des dessins trop petits. De plus, pour éviter des problèmes de compréhension, il nous a été conseillé de ne présenter qu'un seul décor à la fois plutôt que de submerger l'enfant par de trop nombreuses possibilités. L'enfant voit ainsi directement le décor tel qu'il sera lors de la création de la scène. Le déroulement circulaire des décors présentés un par un et avec une taille maximale est mieux adapté aux caractéristiques des enfants IMC.

Ecran 4 : cahier des charges.



Ecran 4 : version finale.



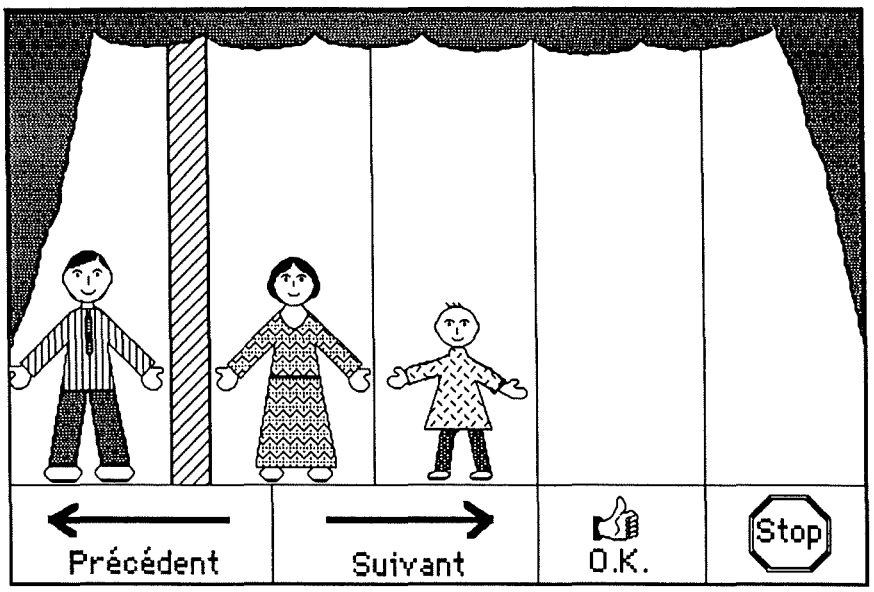
Dans le cahier des charges, les décors prévus sont : pièce de séjour, chambre, forêt (un peu terrifiante), jardin et petite maison, décor féérique et hôpital.

2. choix des marionnettes.

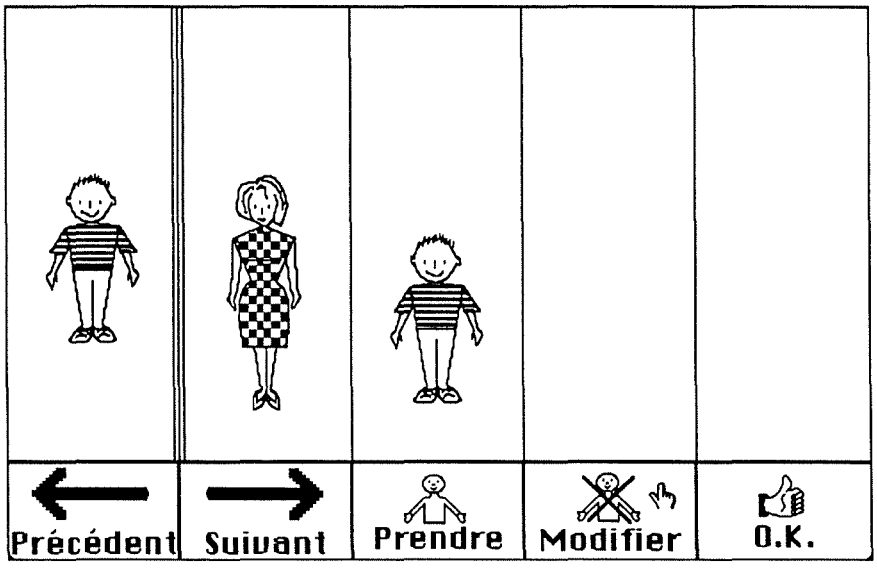
Suite à la sélection de la commande <personnages> de l'écran 3, s'affiche l'écran de choix de ces personnages. L'enfant peut prendre un maximum de 4 marionnettes par scène. Les flèches permettent de visualiser les différents personnages disponibles en les faisant défiler dans la case de gauche. Au fur et à mesure que l'enfant choisit ses marionnettes au moyen de la commande <ok>, celles-ci s'affichent successivement dans les 4 cases de droite. Lorsqu'il a terminé sa sélection, la case <stop> permet le retour à l'écran précédent (écran 3). S'il sélectionne une cinquième marionnette, celle-ci remplace la première choisie. La sixième remplacera la deuxième et ainsi de suite...

Pour des raisons équivalentes à celles exposées dans le choix des décors, un seul personnage à la fois est présenté, contrairement à ce qui était décrit dans la première version de ce cahier des charges.

Ecran 5 : cahier des charges.



Ecran 5 : version finale.



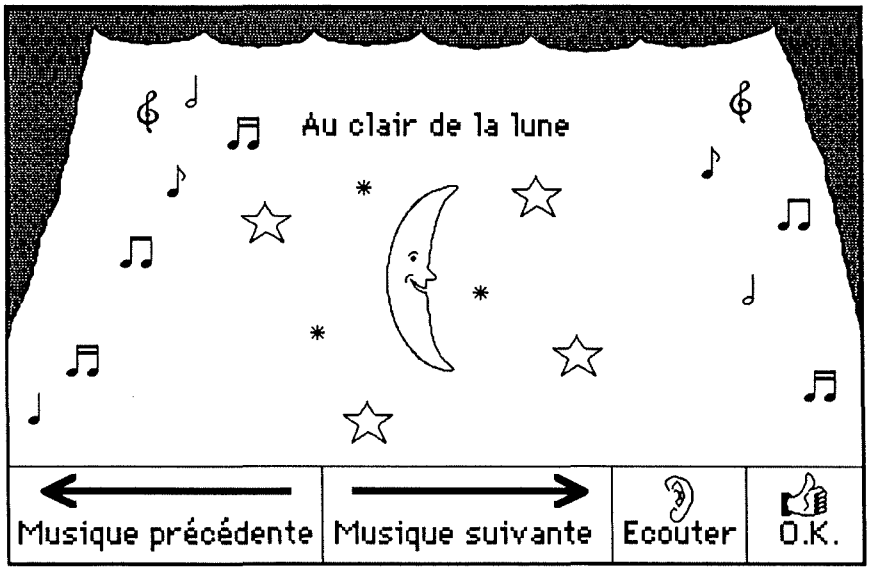
Rappel : les modifications effectuées après la version définitive du cahier des charges seront expliquées par la suite.

Les marionnettes prévues sont : papa, maman, fillette, garçonnet, fée sorcière, policier, infirmier, loup, chien et lapin.

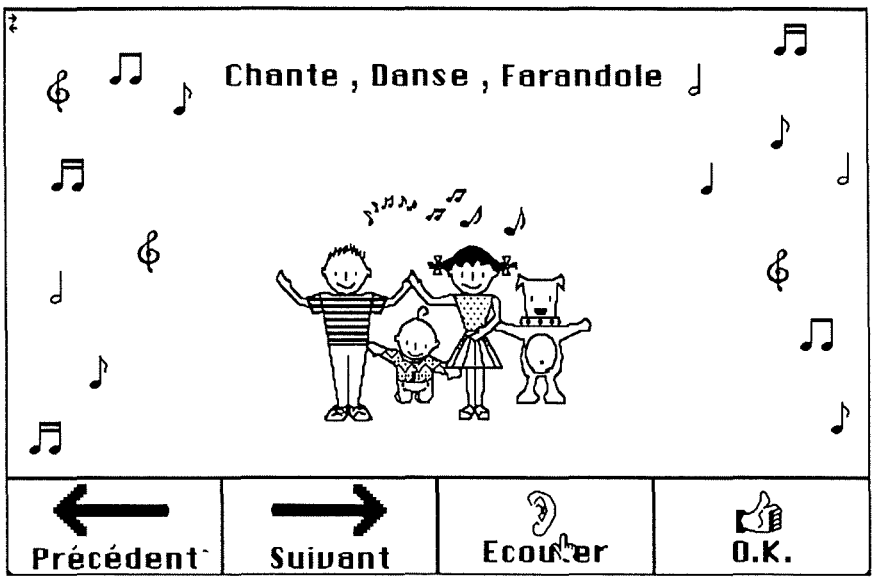
3. Choix de la musique.

La validation de la commande <musique> entraine le passage direct à l'écran de choix de la musique. Celui-ci comprend une illustration simple du thème musical, un titre et 4 commandes: <musique suivante>, <musique précédente>, <écouter> et <ok>. Les deux premières permettent, comme pour le choix du décor et des marionnettes, de passer d'une illustration à l'autre et la troisième offre la possibilité d'entendre la musique. La sélection de la musique s'effectue par la commande <ok> et implique le retour à l'écran 3.

Ecran 6 : cahier des charges.



Ecran 6 : version finale.



Ses sélections terminées, l'enfant passe à la partie d'animation grâce à la commande <ok> de l'écran 3. C'est là qu'il crée sa scène en animant ses marionnettes dans le décor et au son de la musique.

Ceci termine la présentation de l'option "créer une scène". Revenons maintenant à l'écran 2 et explorons la seconde option d'OrdiThéâtre "regarder une scène".

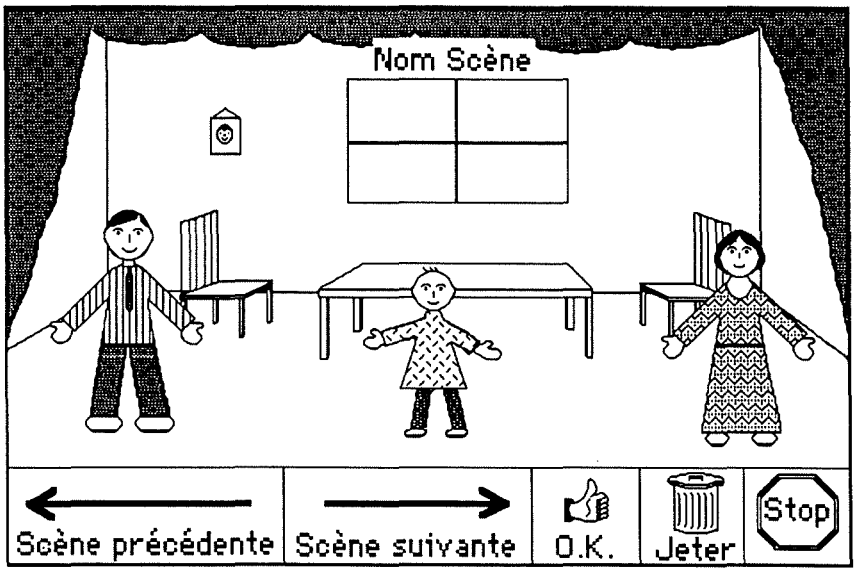
* Option "regarder une scène".

Après avoir créé une scène, l'enfant peut immédiatement aller regarder le résultat de son animation. Pour cela, il sélectionne sur l'écran 2, la commande <regarder une scène>. Dans le cas où aucune scène n'est enregistrée, un message d'erreur apparait et l'on reste à l'écran 2. Ceci laisse la possibilité de changer de disquette ou de choisir l'autre option.

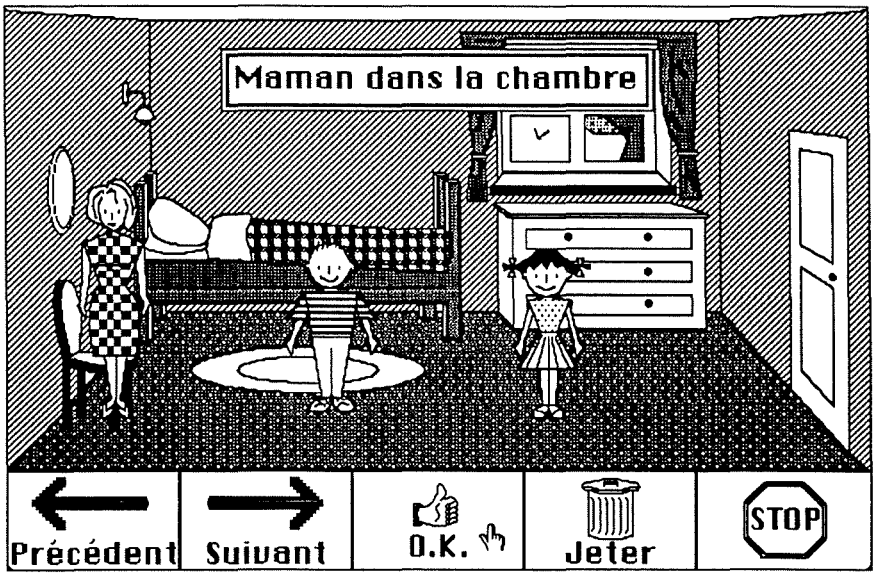
Pour regarder une scène, l'enfant doit en premier lieu choisir celle qu'il désire visualiser. Les scènes enregistrées lui sont proposées une à une et il peut ainsi les faire défiler avant de décider celle qu'il veut. Les scènes sont présentées de la manière suivante : les marionnettes sont

placées en ligne sur le décor et le titre de la scène est affiché en haut de l'écran (pour autant que l'enfant ait donné un titre à sa création).

Ecran 7 : cahier des charges.



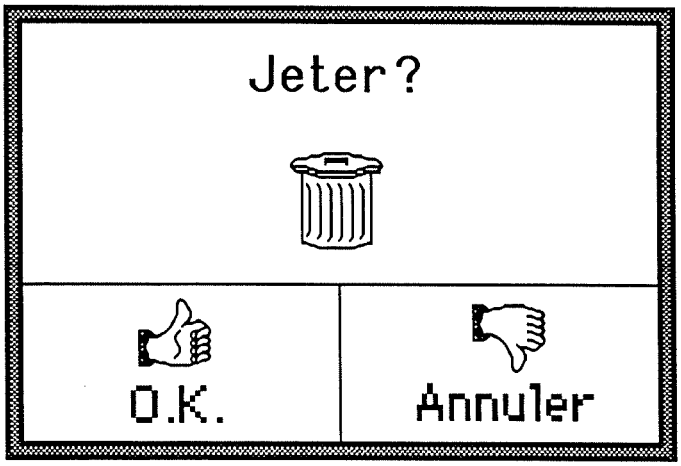
Ecran 7 : version finale.



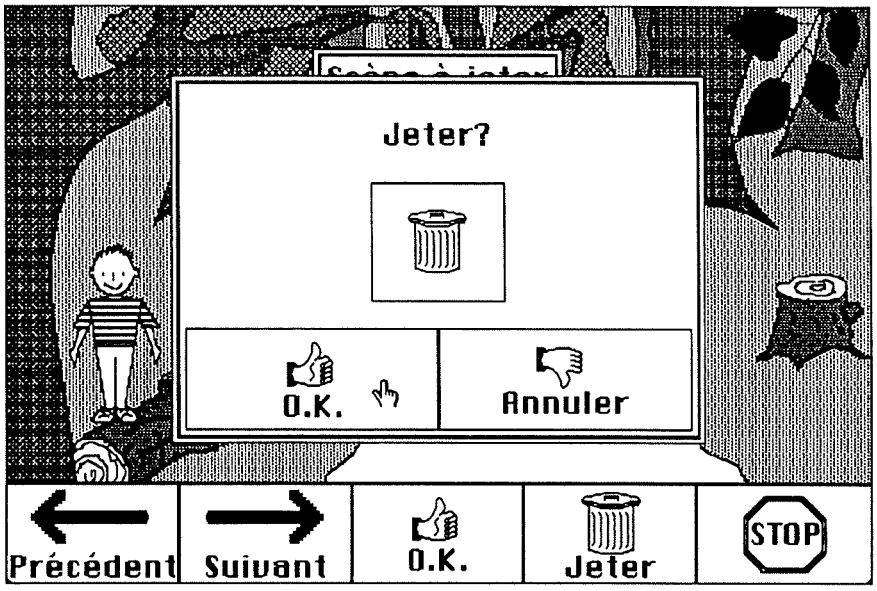
Les flèches permettent de faire défiler les scènes. La commande <jeter>, comme son nom l'indique, permet à l'enfant de jeter une scène qu'il n'aurait plus envie de conserver. Pour éviter des suppressions dues à

des erreurs de manipulation, une confirmation de la commande est demandée à l'enfant.

Fenêtre 2 : cahier des charges.

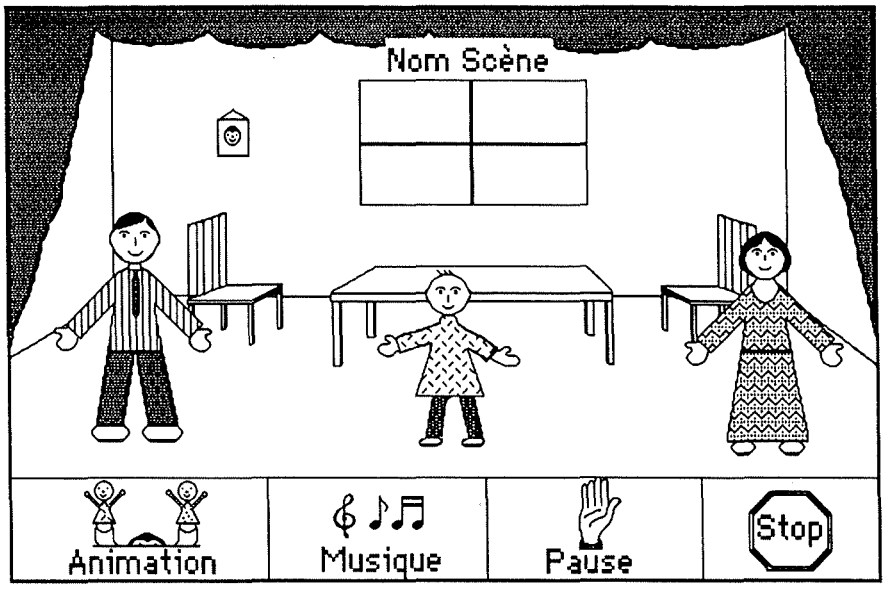


Fenêtre 2 : version finale.

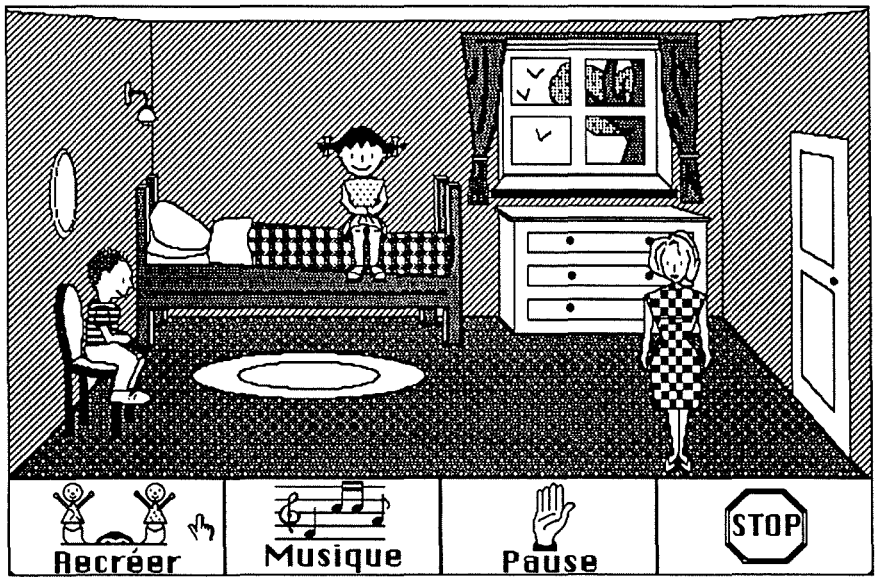


La commande <stop> permet de revenir à l'écran précédent (écran 2). L'enfant effectue son choix par la commande <ok>. Les marionnettes se mettent alors en mouvement, le titre s'efface, la musique se joue et la ligne de commandes est modifiée.

Ecran 8 : cahier des charges.



Ecran 8 : version finale.

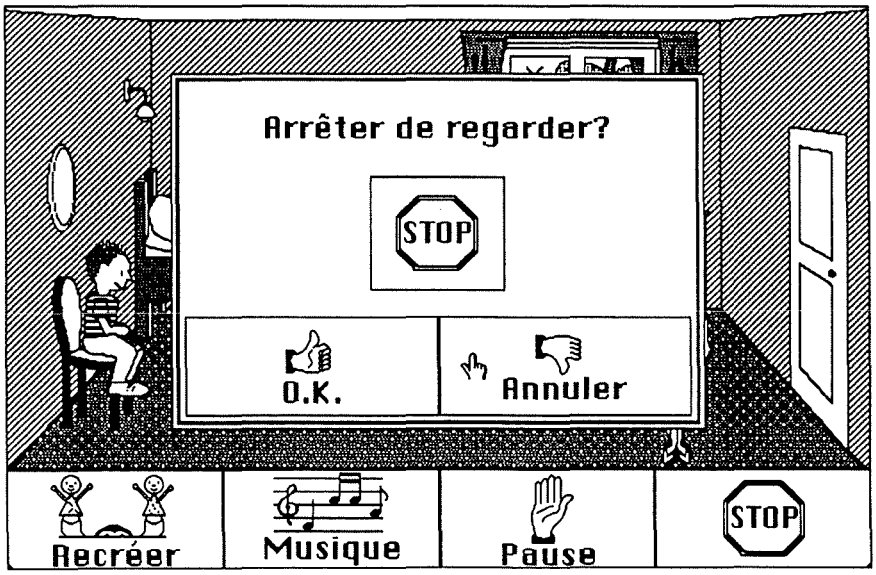


Lorsque la scène se joue, les commandes enregistrées s'exécutent les unes après les autres, sans temps d'attente. Si l'enfant a attendu 10 secondes entre 2 commandes lors de l'enregistrement, celles-ci n'apparaissent lorsque la scène se joue. La commande <pause> permet de

faire un "arrêt sur image". Les marionnettes s'immobilisent et il suffit de sélectionner de nouveau cette commande pour les faire repartir. La case <musique> permet de brancher ou de couper la musique. La case <stop> entraine le retour à l'écran de sélection des scènes à regarder, après une demande de confirmation.

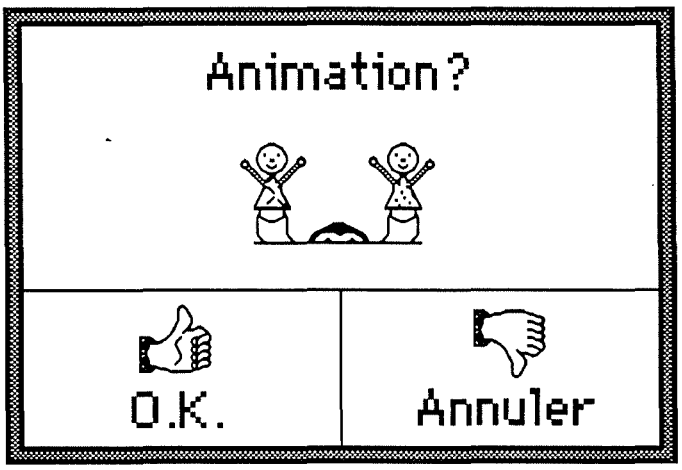
Fenêtre 3 : cahier des charges.

Fenêtre 3 : version finale.

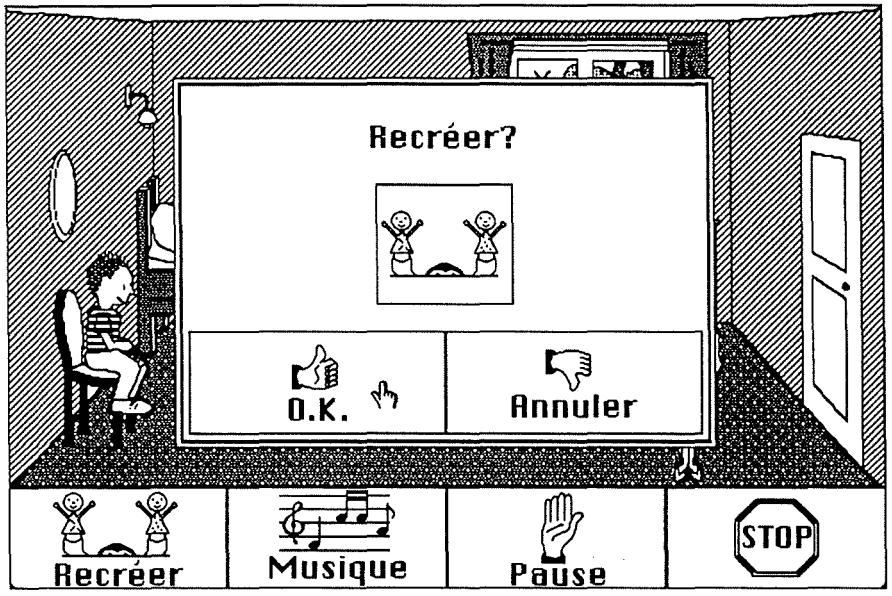


La commande <animation> offre la possibilité d'arrêter la scène et de la recréer à partir de l'endroit où elle s'est arrêtée. Une confirmation est demandée et on se retrouve alors directement dans la partie animation. Si la scène s'était jouée entièrement, cela permet, en branchant l'enregistrement dès l'entrée dans la partie animation, de la continuer sans modifier ce qui avait déjà été réalisé. Si la scène ne s'était pas entièrement déroulée et que l'enfant branche l'enregistrement, tout ce qui n'avait pas été visualisé est effacé et remplacé par la nouvelle création.

Fenêtre 4 : cahier des charges.



Fenêtre 4 : version finale.



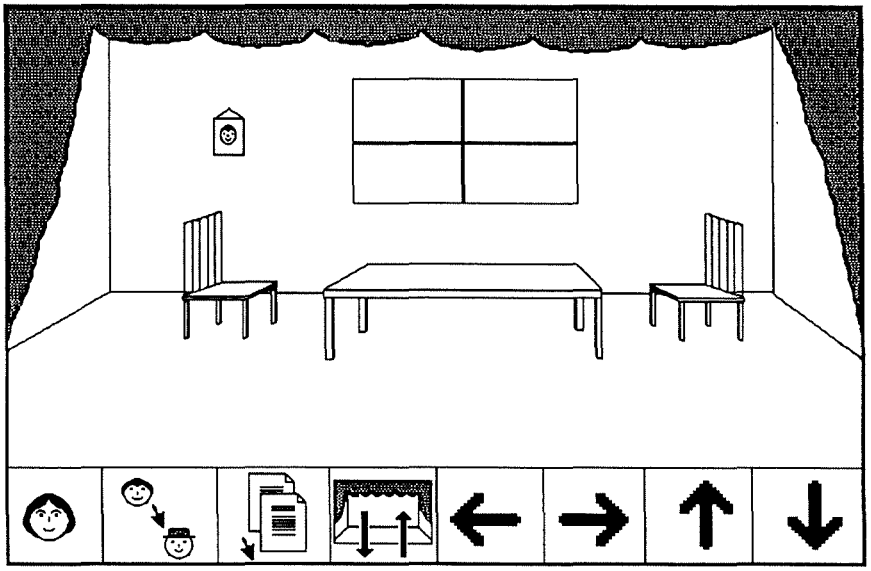
Partie 3.

Cette partie débute par l'affichage du décor choisi par l'enfant, les marionnettes se trouvant "en coulisses". C'est à l'enfant de les faire entrer en scène au moyen des commandes situées dans le bas de l'écran. Le principe de l'animation est simple : l'enfant manipule un personnage à la fois, celui dont la tête est affichée dans la première case (en bas à gauche de l'écran). Ce personnage effectue la commande dès la sélection de celle-ci. Une commande particulière permet de changer de marionnette c'est-à-dire de changer la tête affichée dans la première case. Vu leur nombre, les commandes illustrées par un dessin simple sont réparties sur un maximum de 4 lignes que l'enfant fait apparaitre successivement en sélectionnant la case appropriée.

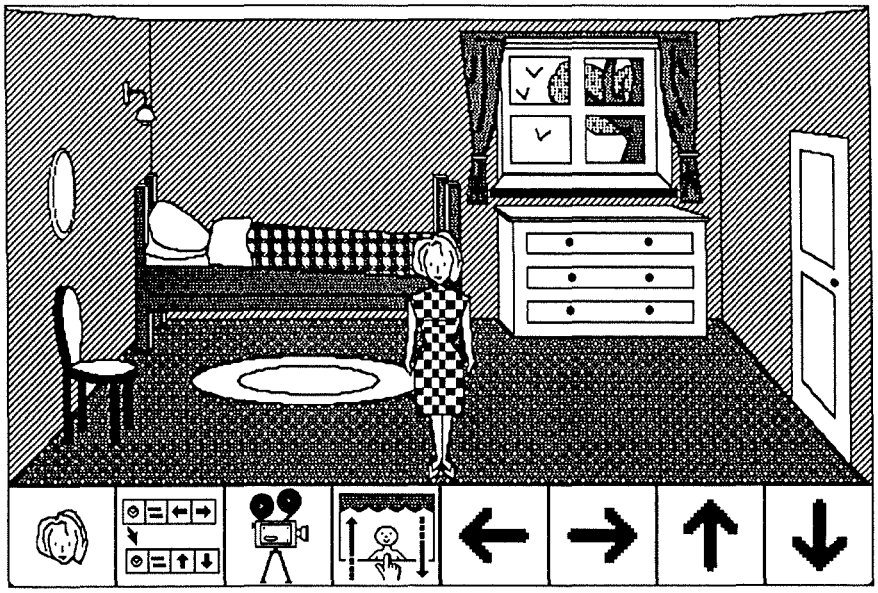
La première version de ce cahier des charges prévoyait une ligne supplémentaire comprenant les 4 marionnettes. L'enfant en sélectionnait une, ce qui impliquait l'apparition des commandes d'animation à la place des autres marionnettes. Pour changer de personnage, il devait revenir à cette ligne en sélectionnant la case contenant le personnage en animation. Ce système impliquait donc une ligne en plus des lignes comprenant les commandes de jeu. Pour ne pas surcharger l'enfant par un nombre trop

élevé de lignes et risquer des problèmes pour la compréhension, il nous a paru plus simple de faire défiler la tête des marionnettes dans la première case à l'aide d'une commande spéciale.

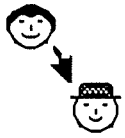
Ecran 9 : cahier des charges.



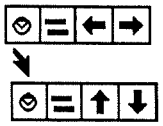
Ecran 9 : version finale.



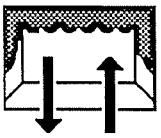
Les commandes sont :

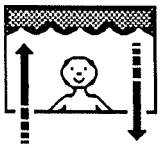
version cahier des charges	version finale
	commande supprimée.

permet de changer la tête de la marionnette présente dans la case de gauche (à condition que l'enfant ait choisi plus d'une marionnette), c'est-à-dire la marionnette qui effectue les commandes sélectionnées.



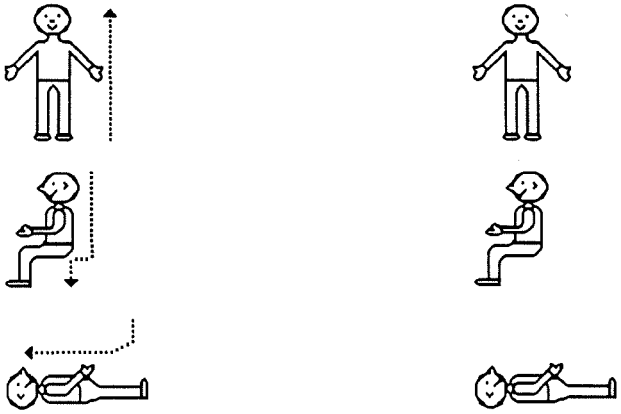
entraine le passage à la ligne de commandes suivante.





fait entrer la marionnette en scène ou la fait sortir.

la marionnette se déplace sur une distance fixe, dans la direction indiquée.



permet de lever, d'asseoir ou de coucher la marionnette.

le personnage tourne d'1/4 de tour vers la gauche ou vers la droite.

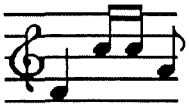
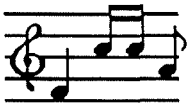
commande supprimée

commande supprimée

permet de modifier l'expression du personnage (gai ou triste).

la marionnette agite les bras.

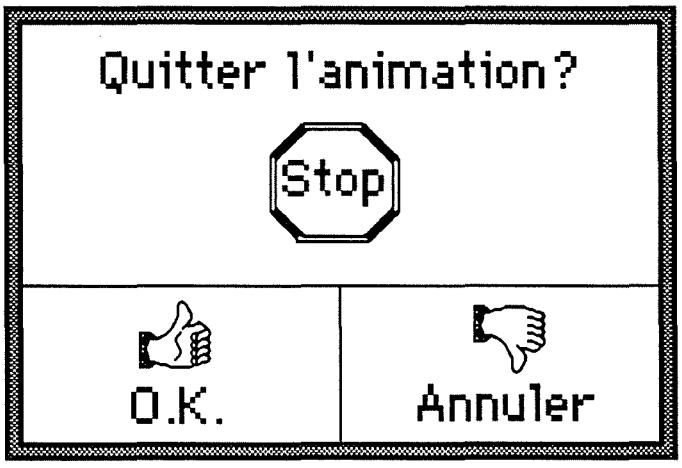

permet de brancher ou de couper l'enregistrement. Quand celui-ci est branché, la case devient plus sombre (elle s'affiche en inverse). Si l'enfant n'enregistre pas en continu, les différents morceaux de scène enregistrés seront mis bout à bout.

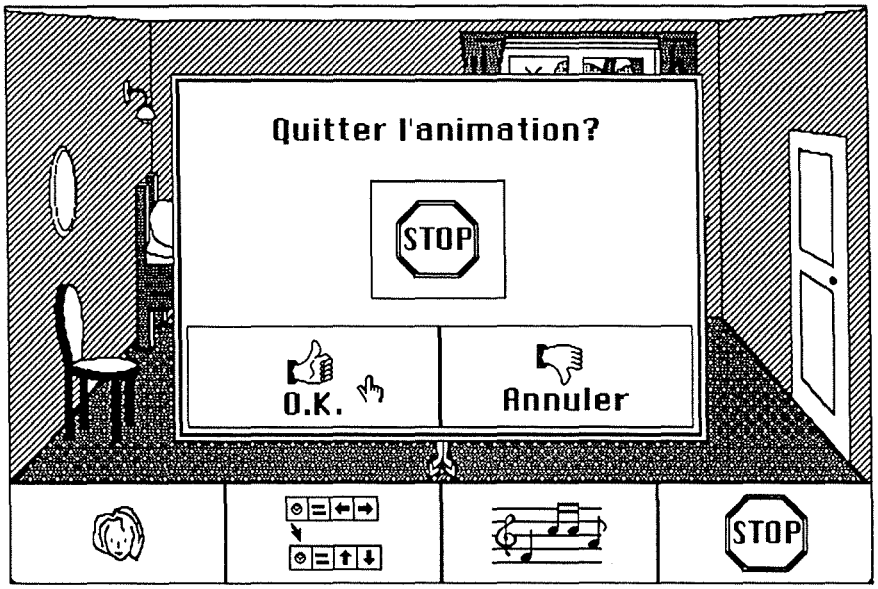
met en route ou coupe la musique.

permet de quitter l'animation et de retourner à l'écran de sélection (écran 3). Pour éviter de quitter l'animation par erreur, une confirmation de la commande est demandée à l'enfant. S'il l'annule, il peut continuer la création de sa scène comme si rien ne s'était produit.

Fenêtre 5 : cahier des charges.

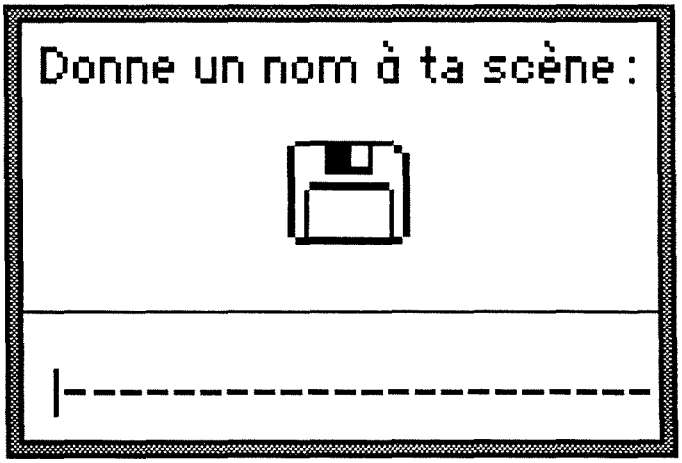


Fenêtre 5 : version finale.



Si l'enfant confirme la commande <quitter> et qu'il a enregistré une scène, il a la possibilité de lui donner un nom.

Fenêtre 6 : cahier des charges.



Fenêtre 6 : version finale.

Ensuite, l'enfant quitte la partie animation et se retrouve à l'écran de sélection (écran 3). Il peut alors aller regarder sa scène en retournant à l'écran de départ (écran 2) et en sélectionnant l'option "regarder une scène" ou bien sélectionner de nouvelles marionnettes et un nouveau décor et recréer une scène.

Ceci conclu la présentation de la version définitive du cahier des charges telle qu'elle fut rédigée fin octobre 1989, au terme de notre séjour dans le service IMC.

2.3. La programmation d'OrdiThéâtre.

Dès novembre, nous avons commencé le travail de conception du logiciel et la programmation d'une partie exécutable de celui-ci. En fait, il était déjà prévu d'effectuer une première série de tests à Paris dès le mois de janvier. Nous devions donc avoir réalisé, pour janvier, une première version du logiciel, incomplète bien entendu mais exécutable et comprenant les fonctionnalités principales. C'est ainsi qu'après deux mois

et demi de travail intensif, nous nous sommes rendus à Paris pour une période de 15 jours afin de tester le logiciel avec les enfants. Ces tests étaient pour nous d'une importance capitale. Nous n'avions en effet aucune idée de la façon dont les enfants allaient accueillir OrdiThéâtre. De nombreux problèmes auraient pu surgir à ce moment et nous obliger à recommencer une partie non négligeable de notre travail.

Heureusement, OrdiThéâtre a été accueilli avec beaucoup d'enthousiasme, non seulement par les enfants mais également par tout le personnel du service IMC : psychologues, ergothérapeutes, médecins, infirmiers(ères), institutrices... Les modifications à apporter au logiciel se révélèrent peu nombreuses et portèrent sur des problèmes mineurs. Il a été merveilleux pour nous de découvrir la joie qu'apportait le logiciel à de nombreux enfants. Des sourires sur tous les visages, des éclats de rire de la part d'enfants pourtant dans des situations pénibles, des demandes répétées pour jouer avec OrdiThéâtre nous prouvèrent que nous étions sur la bonne voie.

Nous exposons ci-après les divers problèmes rencontrés ainsi que les solutions apportées, de cette période de tests en janvier jusqu'à la version finale en juillet 1990. La version du logiciel avec laquelle nous avons effectué ces premiers tests ne comprenait pas tous les décors prévus ni tous les personnages et aucune musique n'était disponible. La trace n'était pas non plus implémentée. Les marionnettes réalisées alors étaient le papa, la maman, la fille, le garçon, le loup et le petit chien. Le choix des décors était limité à la salle à manger et la forêt. Cependant, cela a suffit d'une part à satisfaire les enfants et d'autre part à nous fournir assez d'indications sur les quelques modifications à effectuer.

Les modes d'utilisation.

Le premier aspect du logiciel sur lequel s'est portée notre attention concerne les modes d'utilisation. La question était de savoir si tous les enfants étaient capables d'accéder facilement à OrdiThéâtre et arrivaient à créer une scène. En travaillant avec plusieurs enfants atteints de handicaps différents, il s'est avéré que les modes d'utilisation souris et clavier-flèches ne posaient aucune difficulté. Le défilement nécessitait un travail

long et fastidieux pour l'enfant mais celui-ci parvenait néanmoins à créer une petite scène. Il nous a été demandé de rajouter un renforcement visuel lors de chaque commande de l'enfant : la case s'affiche un court instant en inverse puis revient à la normale. Cependant, pour l'enfant qui commande le logiciel au moyen d'un seul interrupteur qu'il actionne avec la tête, il n'est pas toujours possible de fixer continuellement l'écran. Le mouvement de la tête vers le haut, vers le bas ou sur le côté empêche le maintien du regard en direction de l'écran. C'est pourquoi, nous avons ajouté dans le cas du défilement un renforcement sonore : un "bip" sonore se produit lors de la sélection d'une commande. Ces deux ajouts sont les seuls changements liés aux modes d'utilisation d'OrdiThéâtre.

Partie 1.

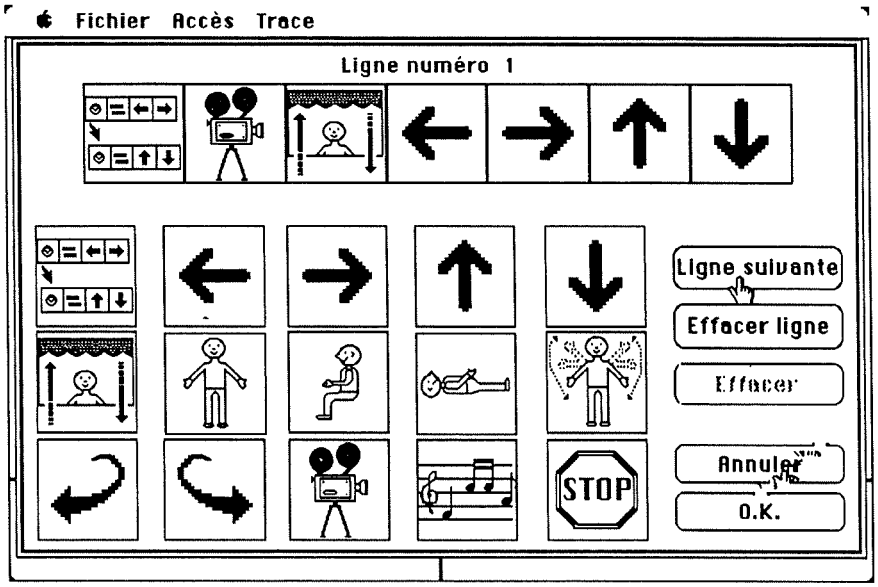
Cette partie, réservée au moniteur, a été considérablement modifiée par l'apport de possibilités supplémentaires.

Nous avons rapidement remarqué qu'il était nécessaire pour l'enfant de procéder par étapes dans la découverte du logiciel. En effet, l'utilisation d'OrdiThéâtre exige un apprentissage, certes faible, mais néanmoins essentiel. La démarche adoptée fut la suivante. Lors de la première séance d'utilisation, l'enfant est placé directement devant l'écran d'animation, le moniteur ayant choisi pour lui un décor, des marionnettes et une musique. L'enfant découvre alors les possibilités d'animation, se prend rapidement au jeu et crée une scène. Lorsqu'il maîtrise cette partie, le moniteur lui explique qu'il a d'autres décors, d'autres marionnettes et d'autres musiques à sa disposition. L'enfant apprend alors la procédure de sélection de ces divers éléments. En dernier lieu, le moniteur révèle à l'enfant la possibilité de regarder les scènes créées.

Pour éviter de noyer certains enfants dans un nombre trop élevé de commandes lors de la première séance, il nous a paru important de permettre au moniteur de décider des commandes proposées à l'enfant ainsi que de leur organisation dans les lignes. Ainsi, le moniteur peut au début n'offrir qu'une seule ligne comprenent la commande de changement de personnage et les quatre flèches de déplacement et, par la suite, ajouter une deuxième ligne permettant d'asseoir, de lever ou de coucher les

marionnettes, et ainsi de suite... Le menu "mode d'utilisation", devenu menu "accès", comprend donc, en plus des trois modes d'utilisation et du réglage de la vitesse de défilement, un élément supplémentaire : l'option <commandes>.

La sélection de cette option entraine l'affichage de la fenêtre suivante:

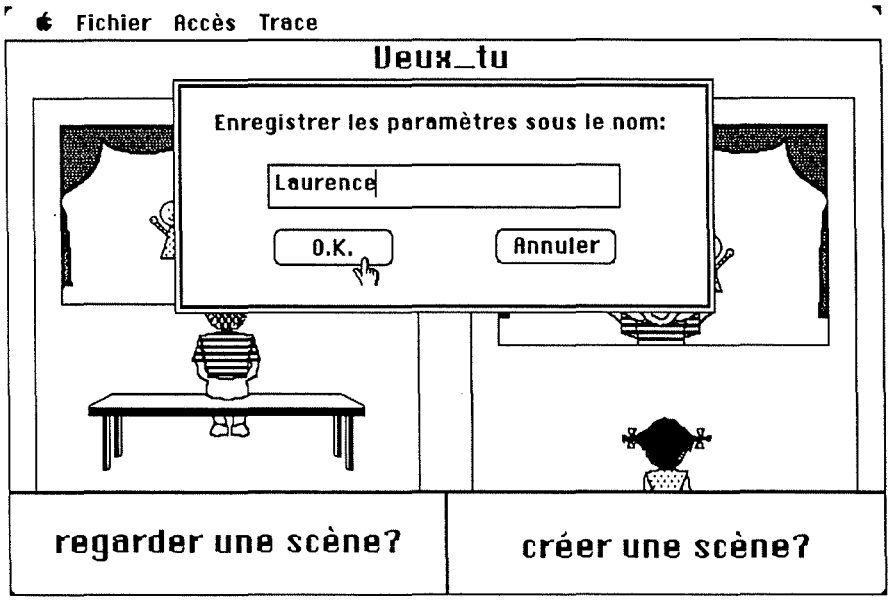


Sur la première ligne apparaissent successivement les 4 lignes de commandes telles qu'elles étaient définies au moment de la sélection de l'option <commandes>. En dessous sont présentées les 15 commandes disponibles. Le moniteur construit les lignes de commandes et les modifie à son gré. Ceci permet l'adaptation du logiciel au niveau de compréhension atteint par l'enfant.

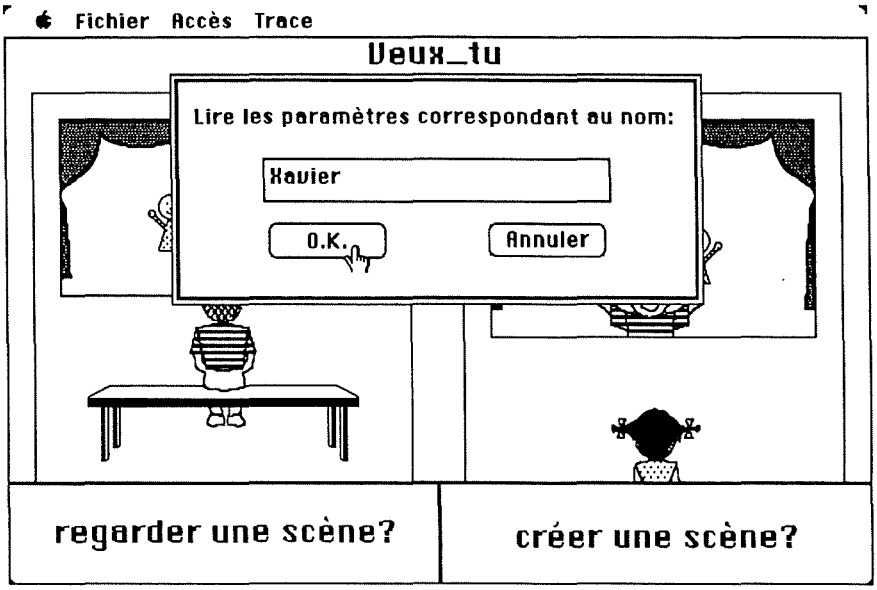
Lors des premières utilisations, l'enfant n'est pas conscient de l'importance de brancher la caméra pour enregistrer son animation et la revoir par la suite. De plus, il n'est pas à l'abris d'un oubli même s'il est habitué au logiciel. C'est pourquoi, nous avons ajouté pour le moniteur une option d'<enregistrement automatique>. Lorsque celui-ci est branché, la caméra enregistre dès l'entrée dans la partie animation. L'enfant peut alors débrancher la caméra et arrêter l'enregistrement volontairement. Cette option se situe dans le menu "fichier" qui remplace le menu "jeu" prévu dans le cahier des charges. Ce menu comprend également l'option

<quitter> ainsi que 2 nouvelles possibilités : la lecture et l'enregistrement des paramètres. En effet, vu l'augmentation du nombre de paramètres à définir, il nous a semblé utile de permettre la mémorisation par le logiciel des paramètres en fonction des enfants. Grâce à l'option <enregistrer les paramètres>, le moniteur adapte le logiciel à l'enfant et enregistre ses caractéristiques selon son nom. Lors de l'utilisation suivante, il suffit d'utiliser l'option <lire les paramètres> et le logiciel va chercher les éléments mémorisés au nom de l'enfant.

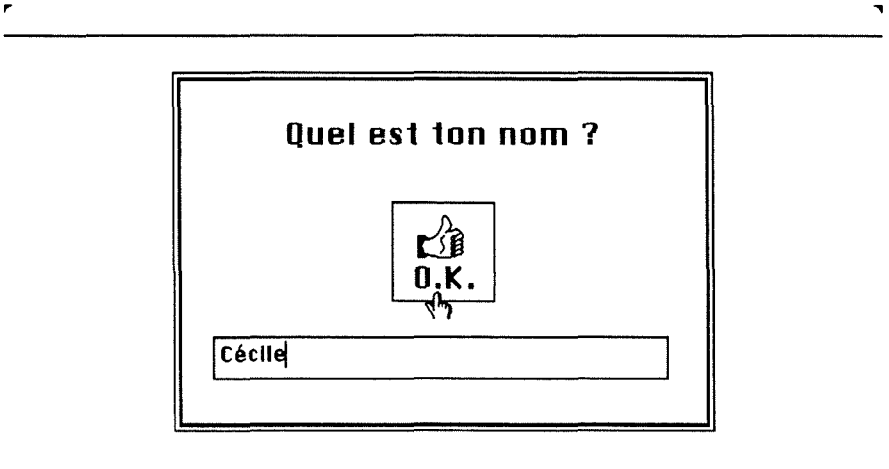
L'option <enregistrer les paramètres> entraine l'apparition de la fenêtre suivante :



L'option <lire les paramètres> est suivie de la fenêtre suivante :



L'option <commencer> nous est apparue superflue. En effet, le moniteur peut de toutes façons accéder aux menus à tout moment durant l'utilisation d'OrdiThéâtre. Il n'est donc pas nécessaire de prévoir un écran vide dans le but d'attendre le signal <commencer> du moniteur. De plus, s'il a lancé le OrdiThéâtre, normalement, c'est qu'il veut l'utiliser. Ainsi, dès la sélection du programme un écran de présentation apparait, suivi de la fenêtre suivante :



Si un nom est introduit, OrdiThéâtre va chercher les paramètres enregistrés correspondants à ce nom, sinon il prend les paramètres définis par défaut. Ensuite, on passe directement à l'écran présentant les deux grandes options du logiciel et l'on entre dans la deuxième partie.

Le menu "trace" est resté inchangé. Un nouveau menu a été ajouté aux trois précédents : le menu "". La première option s'intitule <A propos d'OrdiThéâtre...> et fournit des commentaires généraux concernant le logiciel. Les options suivantes sont des accessoires de bureau Macintosh. Ces accessoires ne sont pas indispensables pour l'utilisation d'OrdiThéâtre. C'est pourquoi ils ne sont pas détaillés ici. Pour obtenir plus de renseignements à ce sujet, il faut se référer au manuel d'utilisation Macintosh.

En résumé, la version définitive d'OrdiThéâtre comprend 4 menus dont les options sont respectivement les suivantes :

: A propos d'OrdiThéâtre...
Accessoires de bureau.

Fichier: Lire les paramètres
Enregistrer les paramètres
Enregistrement automatique
Quitter

Accès: Souris
Clavier-flèches
Clavier-défilement
Vitesse de défilement
Commandes

Trace: Phase de sélection
Phase d'animation
Phase de sélection et phase d'animation
Aucune

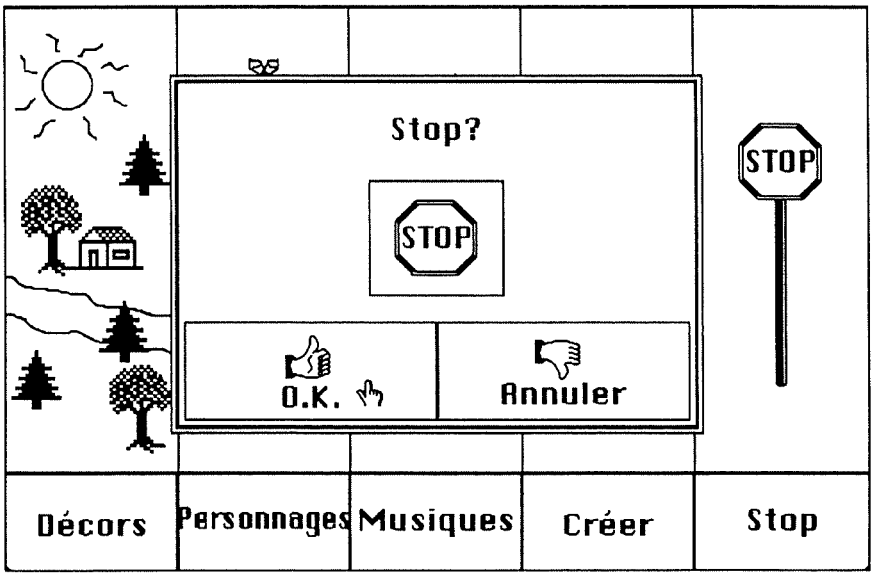
Partie 2.

Cette partie débute par l'écran de présentation des deux options <regarder une scène> et <créer une scène>. Comparé au cahier des charges, aucune modification n'a été apportée à l'écran.

* Option <créer une scène>.

L'unique changement sur l'écran de présentation des divers choix possibles (décors, personnages, musiques) est la modification du terme "ok" en "créer". Ce dernier nous a paru plus explicite et plus facilement compréhensible pour l'enfant.

Etant donné les difficultés d'accès de nombreux enfants à l'ordinateur, des erreurs de manipulation surviennent fréquemment. C'est pourquoi il nous a été demandé d'ajouter des demandes de confirmation suite à certaines commandes. Ainsi la commande <stop> est suivie de la fenêtre suivante :



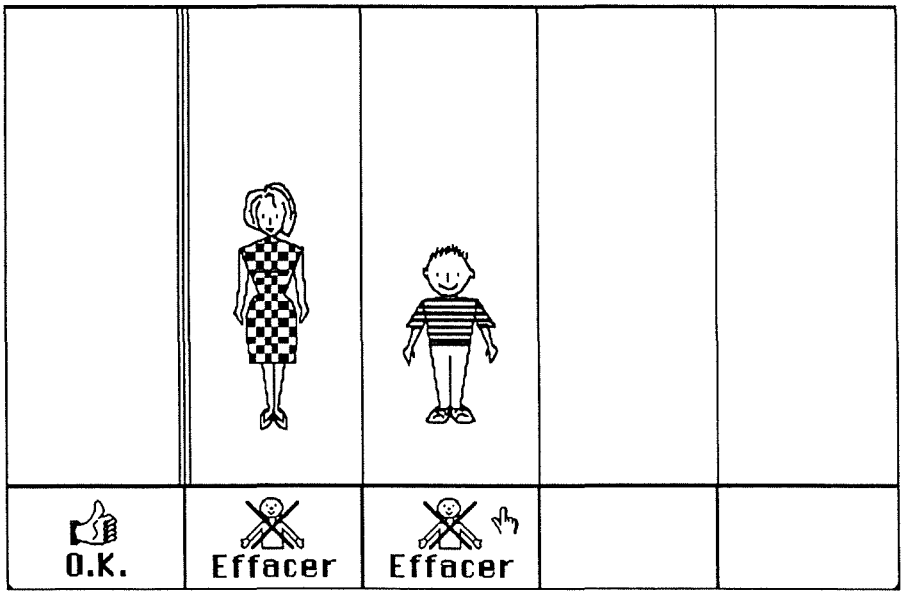
1. Choix du décor.

La technique de choix du décor est restée similaire à celle établie dans le cahier des charges. Pour des raisons d'uniformité, il nous a semblé préférable d'accompagner les flèches des seuls mots "précédent" et "suivant".

Les décors qui ont été réalisé sont : la salle à manger, la chambre, la forêt, un décor féérique (château) et un décor d'extérieur assez neutre (bâtiments).

2. Choix des marionnettes.

Lors de la conception du programme durant les mois de novembre et décembre, nous avons estimé la procédure de modification des personnages choisis trop compliquée pour un bon nombre d'enfants (le 5ème personnage choisi remplace le premier...). Nous avons donc préféré ajouter un écran supplémentaire en vue de simplifier cette procédure. Ainsi, nous avons introduit une commande modifier dans la ligne de commandes. Sa sélection entraine le passage à l'écran de modification des personnages :



L'enfant doit uniquement valider la ou les commande(s) <effacer> correspondant au(x) personnage(s) qu'il désire supprimer. Ensuite, il sélectionne la commande <ok> et revient à l'écran précédent c'est-à-dire l'écran de choix des personnages. Ce dernier, dans le cahier des charges, comprenait également les commandes <ok> et <stop>. L'enfant indiquait par <ok> qu'il désirait le personnage présenté dans la case de gauche et par <stop> qu'il avait terminé son choix et retournait ainsi à l'écran de sélection du décor, des marionnettes ou de la musique. Or pour le choix du décor et de la musique, c'est la commande <ok> qui indiquait le choix et entrainait le retour à l'écran précédent. Pour garder une uniformité dans la signification des commandes, nous avons utilisé la commande <ok> à la place du <stop> prévu dans le cahier des charges et remplacé le <ok> par une commande <prendre> signifiant donc la sélection de la marionnette présentée. Quand l'enfant a pris 4 marionnettes, la case de gauche devient vide puisqu'il ne peut en choisir plus.

Un problème concernant le choix des marionnettes est survenu lors des tests. La marionnette, une fois sélectionnée par la commande <prendre>, met un certain temps à s'afficher dans la case de droite. Les enfants, impatients et ne voyant rien se produire, sélectionnaient alors de nouveau la commande, ce qui entrainait la double sélection du personnage. Pour faire patienter les enfants et leur montrer qu'OrdiThéâtre "travaille", nous avons affiché sur l'écran, durant les temps d'attente, un petit lapin agitant la patte et les oreilles. Les enfants ont alors rapidement compris que quand le lapin apparaissait, il suffisait d'attendre et ne toucher à rien. Voici ce lapin :

Sur la demande de psychologues et de médecins du service IMC, nous avons quelque peu changé les marionnettes à proposer aux enfants : un bébé ainsi qu'un enfant en chaise roulante ont remplacé le policier, l'infirmier et le lapin prévus.

3. Choix de la musique.

Aucune modification n'a été apportée à cet écran. Nous avons digitalisé six musiques enfantines durant le mois de mai et nous les avons illustrées par un dessin simple.

* Option <regarder une scène>.

L'écran de sélection de la scène que l'enfant veut regarder est identique à celui prévu dans le cahier des charges. Pour éviter que des enfants suppriment des scènes ne leur appartenant pas, l'enfant n'a accès qu'à celles qu'il a créées.

Lorsque l'enfant valide la commande <ok>, il faut un certain temps avant que la scène ne se joue. A ce moment, le petit lapin prévu pour faire patienter l'enfant apparaît.

La ligne des commandes de l'écran où la scène se joue n'a subi qu'une très légère modification : le terme <animation> a été remplacé par <recréer>, celui-ci nous paraissant plus adapté.

Partie 3.

La technique d'animation de la version finale du logiciel est la même que celle établie dans le cahier des charges. Quelques changements ont été apportés aux commandes prévues.

Lors des tests, de nombreux enfants sélectionnaient la case représentant la tête du personnage en action, ne comprenant pas qu'il ne s'agissait pas d'une commande. Dès lors, nous avons supprimé la commande de changement de marionnette, celui-ci s'effectuant maintenant en sélectionnant la première case contenant la tête du personnage animé.

Nous avons aussi remplacé les illustrations des commandes <changement de ligne> et <enregistrement> par des illustrations qui nous semblaient plus adaptées et plus compréhensibles. De plus, les commandes

permettant de modifier l'expression des marionnettes a été supprimée car ceci exigeait de trop nombreux dessins (2 fois chaque personnage dans 19 positions différentes !).

Une possibilité supplémentaire a été offerte aux enfants utilisant la souris. Il peuvent en effet déplacer le personnage directement avec la souris plutôt que d'utiliser les commandes de déplacement du bas de l'écran. Il suffit de cliquer une fois sur la marionnette et celle-ci suit le déplacement de la souris. En cliquant de nouveau, la marionnette est libérée et le curseur de la souris réapparait.

Durant le mois de juin, une version presque définitive d'OrdiThéâtre a été testée dans une institution à Lessines. Ce test ayant été concluant, aucune modification n'a été apportée suite à cette période.

Ceci conclut la présentation de la version finale d'OrdiThéâtre. Une explication plus détaillée est exposée dans le manuel d'utilisation fourni en annexe.

3. Analyse de l'interface.

Nous avons, au chapitre 1, énoncé les principes généraux de conception d'une interface homme-machine. Ensuite, au chapitre 3, nous avons tenté, sur base de notre expérience personnelle, d'adapter ces différents principes aux besoins des enfants IMC. Nous allons donc maintenant analyser l'interface d'OrdiThéâtre à la lumière de ces principes.

3.1. Les styles d'interaction utilisés.

Comme nous le conseillons au chapitre 3, la plupart des actions sont effectuées via des menus (les "lignes de commandes"), situés en bas de l'écran. Une ligne de commandes contient différentes "cases" de taille assez grande accolées les unes aux autres, chacunes de celles-ci contenant un item. Il n'est jamais présenté qu'un seul menu à la fois et ceux-ci

s'enchaînent selon une arborescence simple. Les différents items sont toujours représentés soit par un dessin seul, soit par un dessin accompagné d'un texte simple.

Les possibilités de paramétrage de l'interface permettent de manipuler ces menus de trois manières différentes, autorisant ainsi l'utilisation de nombreux matériels d'entrée:

- le premier mode d'utilisation est le mode "souris", dans lequel la sélection d'un item se fait en cliquant dans la case correspondante à l'aide d'un appareil de manipulation directe. Ceci permet d'utiliser notamment une souris, une trackball, un casque ultrasonique ou un joystick.

- le deuxième est le mode "clavier-flèches", dans lequel un curseur peut être déplacé de case en case (vers la gauche ou vers la droite) en utilisant deux touches du clavier. Lorsque ce curseur se trouve sur l'item désiré, une troisième touche (<return>) permet la sélection. Ceci permet une utilisation restreinte du clavier, éventuellement à l'aide d'une licorne, ou d'un jeu de trois interrupteurs.

- le dernier est le mode "clavier-défilement", semblable au précédent mis à part que le curseur se déplace automatiquement de la gauche vers la droite à une vitesse réglable. Lorsque le curseur a atteint l'item désiré, la sélection est effectuée en appuyant sur une seule touche (<return>). Ceci permet l'utilisation d'une seule touche du clavier ou d'un seul interrupteur.

La manipulation directe est également utilisée, de manière très restreinte et facultative. Elle permet une manipulation plus aisée des marionnettes lors de la création d'une scène, mais elle n'est possible que dans le cadre du mode d'utilisation "souris" et n'est jamais obligatoire. En effet, elle n'offre que des possibilités de déplacement des personnages; possibilités par ailleurs offertes dans les lignes de commandes.

3.2. Respect des règles d'or de la conception.

Comme nous l'avons dit précédemment, les règles d'or décrites par Ben Shneiderman sont aussi d'application lorsque l'on développe une interface pour enfants IMC, mais leur importance relative nous paraît différente. Nous allons voir de quelle manière nous avons respecté ces règles d'or.

* La possibilité d'employer des raccourcis.

La possibilité d'employer des raccourcis n'a pas été donnée à l'utilisateur. Comme nous l'avions indiqué, ceci nous semble moins important pour des enfants IMC. En effet, cette règle fait appel à la notion d'*utilisateur expérimenté,* ce que nous jugeons sortir quelque peu du contexte "enfants IMC". De plus, les différentes actions possibles dans OrdiThéâtre sont relativement peu nombreuses et assez rapides à accéder. La présence de raccourcis aurait même peut-être pu perturber l'enfant ou être une source d'erreurs. En conséquence, nous avons préféré ne pas suivre cette règle de conception.

* Le contrôle explicite de l'application par l'utilisateur.

Comme nous avions à développer un logiciel de jeu dont l'un des buts principaux était de permettre à l'enfant de s'exprimer, nous nous sommes attachés à donner le contrôle de l'application à l'utilisateur. Le programme attend constamment une action de l'enfant, qui peut à tout moment arrêter l'opération en cours (choix de personnages ou visualisation d'une scène, par exemple) pour en effectuer une autre. Les sélections des marionnettes, musiques ou décors peuvent également être réalisées dans un ordre quelconque.

* L'organisation de séquences d'actions.

Les différentes actions sont organisées en séquences ayant un début et une fin. Ainsi, la sélection des personnages commence lorsque l'on active la commande "personnages", continue par la visualisation et le choix des différentes marionnettes, puis se termine en activant la commande "stop". Ce modèle de séquence est le même pour toutes les actions du logiciel (choix de personnages, d'un décor, d'une musique ou d'une scène; correction du choix des personnages; créer ou regarder une scène).

* La réduction de la charge de mémoire à court terme.

La charge de mémoire à court terme est quasiment nulle. Lors de la sélection des personnages, les marionnettes déjà choisies sont présentées à l'écran de manière à ce que l'utilisateur ne soit pas obligé de les mémoriser. Si l'enfant lance la création d'une scène en ayant oublié de sélectionner un décor, une musique ou des personnages, le programme le lui signale et lui permet de revenir en arrière pour réparer cet oubli.

* L'uniformité de l'interface.

Nous avons essayé de maintenir un maximum d'uniformité au niveau de l'interface. Toutes les actions se font via la "ligne de commandes" (la zone du bas de l'écran qui comporte les menus destinés à l'enfant), à l'exception de la manipulation directe des personnages lors de l'animation. Toutes les commandes représentées par un même texte ou dessin produisent un résultat équivalent et sont autant que possible situées au même endroit. Ainsi, la commande "Stop" provoque toujours l'arrêt de l'action en cours et le retour à l'écran précédent; elle est chaque fois placée à la fin du menu dont elle fait partie. Cependant, l'organisation des commandes destinées à animer les marionnettes lors de la création d'une scène est laissée aux bons soins du moniteur. Ceci offre plus de souplesse et permet de mieux adapter le logiciel aux différents enfants, mais peut entraîner, si l'on n'y prend garde, un certain manque d'uniformité.

Les actions de choix de personnages, d'un décor, d'une musique ou d'une scène enregistrée, fort semblables, sont traitées de manière similaire: l'utilisateur fait défiler les objets un à un à l'aide des commandes "Précédent" et "Suivant" et indique son accord par la commande "O.K.".

Les fenêtres de dialogue destinées à l'enfant sont regroupées en deux "classes": les fenêtres de confirmation (veux-tu vraiment effacer cette scène...) et les fenêtres de saisie d'une chaîne de caractères (saisie du nom de l'enfant ou du titre d'une scène). Toutes les fenêtres d'une même classe sont semblables, seuls l'image et le texte d'explication changent. La présentation des deux types de fenêtres est différente, ce qui aide à déterminer le genre d'action demandé.

* Le feedback informatif.

Le feedback informatif est principalement visuel de manière à ne pas provoquer d'interférences avec les thèmes musicaux. Lorsque l'on sélectionne un item dans une ligne de commandes, la "case" le contenant s'affiche en vidéo inverse jusqu'à ce que le contacteur employé (bouton de la souris, touche du clavier, interrupteur...) soit relâché. Ces cases étant relativement grandes, ce procédé indique assez clairement à l'utilisateur que son action a été prise en compte. Un feedback sonore est ajouté dans le cas du défilement car l'enfant commandant le logiciel de la tête est dans l'impossibilité de fixer constamment l'écran.

Lorsque le temps de réponse du logiciel est long, comme par exemple lorsque le programme va chercher un personnage sur le disque, il est également utile de fournir un feedback particulier indiquant qu'un temps d'attente est nécessaire. Dans OrdiThéâtre, il se présente sous la forme d'un petit lapin venant agiter la patte au centre de l'écran afin de faire patienter l'utilisateur.

* La gestion des erreurs.

L'ensemble des manipulations se faisant par le biais de menus, les erreurs possibles se ramènent toutes à l'activation malencontreuse d'une commande. Il est donc très facile de prévoir les problèmes graves afin d'essayer de les éviter. La gestion des erreurs suit donc une logique préventive: à chaque fois qu'est effectuée une action importante, risquant de provoquer une perte irrécupérable (effacement d'une scène enregistrée) ou un quelconque désagrément (arrêt involontaire de la scène en cours de création, par exemple), une fenêtre de confirmation apparaît à l'écran. Celle-ci a pour but d'indiquer le caractère important de l'action et d'offrir la possibilité d'y renoncer.

* Le retour en arrière.

A chaque étape du logiciel, la ligne de commandes contient un item "Stop" permettant le retour à l'écran précédent. Il est ainsi permis à l'utilisateur de voyager facilement dans le programme en essayant les différentes options possibles. Cependant, l'annulation pure et simple de la dernière commande n'a été fournie, par le biais de fenêtres de confirmation, que dans le cas d'actions importantes (voir gestion des erreurs). Ceci permet à l'enfant de tester sans risques les options inconnues, tout en évitant qu'un usage inconsidéré d'une commande d'annulation mal comprise ne provoque des erreurs supplémentaires.

L'analyse de l'interface d'OrdiThéâtre effectuée, il nous faut encore évaluer le logiciel afin de voir s'il remplit les objectifs présentés au début de ce chapitre. Nous exposons dans le point suivant les résultats de cette évaluation.

4. Evaluation.

Durant 10 mois, nous nous sommes efforcés de créer un logiciel d'une part, attrayant et adapté aux enfants IMC et d'autre part, répondant

aux demandes des moniteurs travaillant avec les enfants. Il est important ensuite de connaître le résultat de ce travail: les enfants ont-ils compris OrdiThéâtre; comment l'ont-ils perçus; quelles ont été leurs réactions? Les buts de ce logiciel, décrits dans le point 1, ont-ils été atteints? Pour répondre à ces questions, une évaluation complète d'OrdiThéâtre doit être menée. Malheureusement, le logiciel ayant été terminé en juillet, nous n'avons pas eu le temps de l'effectuer. Cependant, nous avons pu, dès le mois de janvier, lancer une évaluation concernant une version incomplète d'OrdiThéâtre. Celui-ci ne comprenait qu'une partie des décors et des marionnettes prévues et ne disposait d'aucune musique. Nous présentons ci-après, les résultats de cette évaluation partielle du logiciel.

Pour réaliser cette évaluation, nous avons distribué des questionnaires aux moniteurs travaillant avec les enfants. Ils contenaient deux types de questions : des questions générales sur les réactions de l'enfant et des questions précises, par écran, concernant la compréhension des différentes commandes et autres éléments de l'écran. Chaque enfant faisant partie de l'évaluation a travaillé avec OrdiThéâtre durant 5 séances. A chacune d'entre elles, le moniteur remplissait le questionnaire, ce qui a permis de mettre en évidence les progrès de l'enfant au fur et à mesure de son travail. Trois stades d'expérimentation correspondant au niveau de compréhension atteint par l'enfant ont été définis, permettant une découverte progressive du logiciel. A la fin de chaque séance, le moniteur indiquait le stade atteint par l'enfant.

Stade 1.

On place l'enfant directement devant l'écran d'animation. Celui-ci ne comprend pas de décor. Il n'y a que deux personnages disponibles (la petite fille et le petit garçon), placés au milieu de l'écran .Deux lignes de commandes sont accessibles :

1:(tête du personnage)// chgt de ligne // entrée,sortie scène // gauche // droite // haut // bas.

2: (tête du personnage)// chgt de ligne // debout // assis // couché // agiter // tourner // stop.

Attention à ne pas oublier de brancher l'enregistrement automatique (voir menu 'fichier') avant d'accéder à l'écran d'animation. Lorsque l'enfant maîtrise ces commandes, on passe au stade 2.

Stade 2.

On montre à l'enfant qu'il existe d'autres décors et d'autres marionnettes. On lui laisse la possibilité de choisir un décor et des marionnettes et ensuite de créer sa scène. De plus, on débranche l'enregistrement automatique et on ajoute la caméra sur la première ligne de commandes. A l'enfant de décider lui-même s'il veut enregistrer ce qu'il crée. Lorsqu'il maîtrise les choix des personnages, du décor et l'animation, on passe au stade suivant.

Stade 3.

On place l'enfant devant l'écran de départ ('Veux-tu regarder...'). On lui explique qu'il peut regarder une scène qu'il a créée précédemment, qu'il peut la continuer ou en créer une nouvelle.

Onze enfants, 6 filles et 5 garçons, ont participé à cette évaluation. Leurs âges vont de 5 à 11 ans (1 enfant de 5 ans, 5 de 6 ans, 2 de 9 ans et 3 de 11 ans). Quatre d'entre eux (les plus grands) savent lire et écrire, 5 savent uniquement lire et écrire leur nom et quelques mots et 2 sont incapables de lire et d'écrire. Ces enfants ont eu peu ou pas de contact avec un ordinateur auparavant. Cinq d'entre eux souffrent de troubles leur rendant la parole impossible ou presque incompréhensible.

4.1. Résultats des questions par écran.

Ecran d'animation.

Tous les enfants ont compris, plus ou moins rapidement, la technique d'animation et les différentes commandes de cet écran. Les deux commandes les plus difficilement comprises sont la caméra et le changement de ligne de commandes. Huit enfants sont moyennement ou ne sont pas perturbés par la multiplicité des lignes de commandes tandis que les 3 autres le sont fortement. Le fait de ne pas pouvoir faire passer les marionnettes derrière les objets du décor, celui-ci étant une toile de fond, ne trouble aucun des enfants, de la première à la dernière séance. Ceci est confirmé par le fait qu'aucun enfant n'essaie de déplacer les objets du décor. La possibilité de déplacer les marionnettes directement avec la souris est utilisée de façon très variable selon les enfants et selon les séances. Deux enfants donnent systématiquement un nom à la scène créée tandis que 2 autres n'en donnent jamais. Pour les 7 enfants restants, cela dépend d'une séance à l'autre, en étant plus fréquent au fur et à mesure des utilisations.

En résumé, l'animation des marionnettes ne pose pas de difficulté aux enfants, même aux plus jeunes. Elle ne requiert que pour certains un peu d'aide pour la compréhension et la maîtrise de quelques commandes.

Ecran de choix des personnages, du décor et de la musique.

Tous les enfants semblent avoir du mal à comprendre la commande <stop> et quelle est sa fonction. La commande <créer> est comprise difficilement par les plus grands et nécessite toujours, pour les plus petits, une aide répétée même après 5 séances. Par contre, pour les commandes <décor> et <personnages>, la plupart des enfants n'ont plus besoin d'explication lors de la dernière séance de travail. Ils arrivent tous progressivement à comprendre qu'ils doivent sélectionner un décor et des marionnettes avant de créer une scène même si quelques uns requièrent encore une explication minimale.

Les fonctions des commandes de cet écran ne semblent pas être complètement assimilées par tous les enfants. En fait, cet écran demande une compréhension plus globale du logiciel que l'écran d'animation. Dans ce dernier, il suffisait de comprendre comment animer les marionnettes sur l'écran tandis qu'ici, il faut assimiler tout le travail de préparation de la scène : choisir un décor, choisir des marionnettes et ensuite, aller créer une scène. L'acquisition de cette vue plus globale du fonctionnement d'OrdiThéâtre nécessite donc un apprentissage plus important, surtout pour les tout petits.

Ecran de choix du décor.

Il ressort des questionnaires que la commande la plus difficile est <précédent>. Cinq enfants sur les 11 ont encore besoin, après les 5 séances, d'une aide répétée en ce qui concerne cette commande. Les autres commandes ne posent pas de problèmes. Aucun enfant ne prend le premier décor qui lui est présenté sans regarder les autres et il n'y a pas de préférence marquée pour un décor plutôt qu'un autre.

Le processus de choix du décor, assez simple, est vite maîtrisé par les enfants.

Ecran de choix des personnages.

A l'exception de la commande <précédent>, les commandes de cet écran ont été relativement bien comprises. Comme pour les décors, tous les enfants font défiler plusieurs marionnettes avant de choisir. Ils comprennent qu'ils peuvent sélectionner entre 1 et 4 marionnettes et ils utilisent rarement la possibilité de modifier leur choix. Les personnages les plus souvent sélectionnés sont la maman, le papa, la fille, le garçon ainsi que le loup qui est fortement préféré au chien.

Ecran de modification du choix des personnages.

Des quelques accès effectués à cet écran, il ressort que les 5 enfants âgés de 9 à 11 ans n'ont pas eu de difficulté tandis que les petits (5 et 6 ans) n'ont pas vraiment compris son utilité et son fonctionnement. Ceci est sans doute dû aux utilisations peu fréquentes de la commande <modifier>. Quelques séances de travail supplémentaires favoriseraient beaucoup la compréhension de cet écran.

Ecran de départ.

Trois enfants ont toujours besoin d'une explication, même minimale, des deux grandes options du logiciel. Deux d'entre eux ne semblent pas avoir compris l'entièreté d'OrdiThéâtre et le troisième n'a jamais utilisé l'option <regarder une scène>. Il est difficile de dire si c'est parce qu'il n'a pas compris cette option ou parce qu'il ne désire pas l'utiliser. Les 8 autres enfants sont arrivés assez facilement à la compréhension de cet écran.

Apparemment, 2 enfants sont arrivés, sur 5 séances, à comprendre l'animation des marionnettes et la sélection de celles-ci ainsi que du décor mais ils n'ont pas eu le temps d'assimiler le reste du logiciel. Ce qui ne veut pas dire qu'ils en sont incapables mais, simplement, qu'ils ont besoin d'un temps d'apprentissage plus long pour arriver à une maîtrise complète d'OrdiThéâtre.

Ecran de choix de la scène à regarder.

Un enfant n'a jamais utilisé cette option et deux autres ne l'ont pas comprise. Il s'agit des trois même enfants ayant eu des problèmes à l'écran précédent. En ce qui concerne les autres, les commandes <précédent> et <jeter> ont été les plus dures à maîtriser. Un seul d'entre eux choisit systématiquement la première scène qui lui est présentée tandis que les autres en passent au moins quelques unes en revue avant de choisir. Aucun problème marquant n'apparait pour cet écran.

Ecran de visualisation de la scène.

Un quatrième enfant (en plus des trois cités précédemment) semble s'être arrêté là dans compréhension d'OrdiThéâtre. Il n'a compris aucune des commandes de cet écran. Deux autres se sont contentés de regarder la scène se jouer et n'ont pas utilisé les commandes. Il n'est donc pas aisé de dire s'ils les ont comprises. Les 5 autres enfants ont compris cet écran assez rapidement même si une explication minimale est encore nécessaire pour certains.

Cette option du logiciel étant découverte en dernier lieu selon la procédure d'évaluation, les enfants ont eu, sur 5 séances, moins de temps pour l'assimiler qu'en ce qui concerne l'animation et le choix du décor et des marionnettes. Il est donc assez normal que les plus petits ne soient pas arrivés à un bon niveau de compréhension de cette partie d'OrdiThéâtre. Il est probable qu'avec quelques séances supplémentaires, les problèmes existants soient supprimés.

4.2. Résultats des questions générales.

Tous les enfants ont rapidement compris qu'ils devaient toujours utiliser la ligne de commandes située dans le bas de l'écran pour manipuler OrdiThéâtre. Les 3 aînés se basent sur les dessins et sur les textes pour se retrouver tandis que les autres ne se réfèrent qu'aux dessins et jamais aux textes. Aucun des 11 enfants n'essaie fréquemment de cliquer sur des objets non interactifs et ils renoncent rarement à leur commande suite à la demande de confirmation. Ceci tend à montrer qu'ils commettent peu d'erreurs de manipulation et qu'ils savent ce qu'ils veulent lorsqu'ils valident des commandes comme le <stop>. Deux enfants seulement placent de temps en temps mal le curseur de la souris à cause de sa forme. Le curseur choisit parait donc adapté. Le temps de réaction du logiciel à certains moments a légèrement perturbé 4 enfants au début mais lors de la dernière séance, cela ne posait plus aucun problème. Nous avons remarqué que la commande <suivant> était fortement préférée à la

commande <précédent>, que ce soit pour le choix du décor ou des marionnettes. Peu d'enfants demandent pour revoir les scènes qu'ils ont créées surtout lors des premières utilisations. Enfin, les séances sont interrompues aussi bien par les enfants que par l'adulte. Lorsque les enfants arrêtent, c'est parce qu'ils sont fatigués et qu'ils ne se sentent plus capables de se concentrer. En effet, l'utilisation de l'ordinateur par les IMC leur demande beaucoup d'efforts et ils se fatiguent donc assez rapidement. Les adultes, eux, interrompent les séances pour des raisons comme l'heure des repas ou parce que les enfants ont des rendez-vous ou bien parce que les enfants sont fatigués même s'ils ne veulent pas toujours le reconnaître.

En conclusion, nous pouvons dire qu'OrdiThéâtre est accessible à tous les enfants, y compris aux plus petits (5 et 6 ans) et aux plus handicapés : Julie qui ne peut ni se déplacer, ni manipuler, ni parler en raison de son handicap moteur a réussi à créer une scène grâce au mode d'utilisation défilement. Le premier objectif qui visait à permettre à un maximum d'enfants de jouer à OrdiThéâtre est donc atteint. Au niveau de la compréhension, l'option <créer une scène> n'a pas posé de problème. Elle a nécessité un apprentissage plus important pour les enfants plus jeunes que pour les aînés. C'est pourquoi, sur 5 séances, les plus petits n'ont pas eu le temps de découvrir et apprendre la seconde option <regarder une scène>. Cependant, celle-ci ne devrait pas leur poser de difficulté puisqu'elle n'est pas plus compliquée que la première. Il leur faut simplement quelques séances supplémentaires pour qu'ils aient le temps de l'assimiler. Les grands, par contre, l'ont comprise et maîtrisée facilement.

OrdiThéâtre, dans son entièreté, est donc compréhensible par tous moyennant un apprentissage plus ou moins important selon les enfants. Il reste à voir si tous les enfants sont disposés à fournir cet effort d'apprentissage. Examinons pour cela quelles sont leurs réactions vis-à-vis du logiciel : sont-ils fortement attirés par celui-ci et motivés à le découvrir et à le maîtriser ou ne le trouvent-ils pas intéressant et s'en lassent-ils vite ? Il est intéressant également de connaître les différentes utilisations que les moniteurs ont pu en dégager.

4.3. Réactions des enfants.

Tous les enfants qui savent parler font des commentaires sur tout ce qu'ils voient et sur ce qu'ils font. Le logiciel favorise donc l'expression et la communication puisque les enfants ressentent le besoin d'exprimer à l'adulte des considérations sur tout ce qui se passe. Ils ont manifesté fréquemment du plaisir et ont très rarement montré des signes de lassitude. Ils n'ont pratiquement jamais exprimé de l'insatisfaction et, excepté les 2 aînés, ont très souvent demandé à réutiliser OrdiThéâtre.

Les enfants paraissent donc fortement attirés par OrdiThéâtre et sont ravis de pouvoir y jouer. Le logiciel atteint son objectif d'utilisation ludique et de moyen d'expression. Il reste à voir si les moniteurs en sont satisfaits du point de vue des utilisations pédagogique et thérapeutique qu'ils espéraient pouvoir en dégager. Passons en revue les divers commentaires des moniteurs en fonction des enfants et nous aurons ainsi leur avis à ce sujet.

Virginie et Frédéric (11 ans) ont tous deux pris beaucoup de plaisir à apprendre OrdiThéâtre et à le maîtriser, ce à quoi ils sont arrivés en un temps relativement court. L'intérêt qu'ils ont pour le logiciel est donc essentiellement cognitif. Ils ont passé du temps à affiner leurs connaissances qui n'étaient pas parfaites en explorant les commandes et les opérations qu'ils comprenaient moins bien. Pour ces enfants, c'est le plaisir de l'utilisation de l'ordinateur qui domine et pas vraiment le fait de pouvoir jouer aux marionnettes même s'ils mettaient du soin à créer leurs scènes. Une fois le fonctionnement d'OrdiThéâtre maîtrisé, ils s'en sont un peu désintéressés. Ils se sont cependant projetés dans les scènes qu'ils ont créées et s'en sont servis pour exprimer certaines choses. Il semble donc que le logiciel soit un peu limité pour des enfants de cet âge.

May Tsu (presque 11 ans) ne peut s'exprimer par le langage. Elle a manifesté par le sourire en fin de séance et entre deux séances qu'elle prenait plaisir à jouer avec OrdiThéâtre. Elle a compris rapidement les commandes et a beaucoup apprécié de créer des scènes. Pour May Tsu, l'objectif d'expression a été atteint puisqu'elle ne peut parler et qu'elle a pu confirmer, par les scènes jouées, les idées et les impressions que les moniteurs avaient à son sujet. Dans son cas, une utilisation thérapeutique a

pu être dégagée. OrdiThéâtre est adapté pour développer sa créativité et lui fournir la possibilité d'exprimer ses idées et ses sentiments. Cependant, il semble que, comme pour Virginie et Frédéric, les possibilités du logiciel soient un peu limitées pour l'âge de May Tsu.

Karim (9 ans) demande fréquemment pour jouer à l'ordinateur mais il a du mal à se concentrer de façon suivie, il s'impatiente rapidement avant de chercher à comprendre. Il est cependant arrivé à une certaine maîtrise du logiciel mais son utilisation fut assez décevante en raison de ses troubles de l'attention et du comportement. Karim est donc très attiré par OrdiThéâtre mais le logiciel demande trop d'attention pour cet enfant.

Denis (9 ans) a rapidement su se servir d'OrdiThéâtre avec efficacité. Cependant, il a développé une forte appréhension face au logiciel. Il avait peur sans savoir dire pourquoi. En fait, il avait peur de ce qu'il voyait sur l'écran car c'était lui dans la vie. Pour Denis, qui ne parle pas, le logiciel a permis une observation de l'enfant et une interprétation de ce qu'il ressentait. Ceci remplit bien l'objectif d'être à l'écoute de l'enfant et de détecter ses problèmes éventuels. Une utilisation psychothérapeutique dans ce cas est envisageable.

Julie (6 ans) souffre d'un handicap grave qui lui enlève toute possibilité de parole, de marche et de manipulation. Elle a réussi cependant, grâce à quelques aménagements, à créer des scènes dans lesquelles elle se projetait. Elle a montré par là des éléments de sa problématique psychique qu'elle n'aurait su exprimer autrement en raison de son handicap de communication. Dans son cas, une utilisation psychothérapeutique est possible. OrdiThéâtre lui offre la possibilité d'exprimer ce qui la préoccupe et permet au psychologue de mieux comprendre ses problèmes. Les objectifs du logiciel sont donc complètement atteints dans le cas d'une enfant comme Julie.

Franck (6 ans) a eu beaucoup de mal à comprendre et à maîtriser OrdiThéâtre. Cependant, un certain apprentissage a été réalisé. Malheureusement, l'aspect projectif du logiciel semble angoissant pour lui. Ainsi, pour Franck, OrdiThéâtre parait contre-indiqué.

Angélique (6 ans) a pris beaucoup de plaisir à accéder à l'ordinateur et à comprendre et utiliser OrdiThéâtre. Elle s'est fortement impliquée dans les scènes qu'elle a créées et s'est identifiée aux marionnettes. Elle a exprimé son impuissance d'enfant handicapée motrice et la frustration et l'agressivité qui en résultent. Angélique fait partie des enfants les plus jeunes et elle est parvenue à maîtriser le logiciel, ce qui est un élement important en faveur d'OrdiThéâtre. Pour une enfant comme Angélique, toutes les utilisations espérées sont possibles: psychothérapeutique puisqu'elle s'exprime fortement; ludique puisqu'elle apprécie de créer des scènes et pédagogique car il y a eu un apprentissage sur le plan spatial, on pourrait ainsi travailler avec elle la structuration de l'espace.

Aurélie (6 ans) a eu beaucoup de difficultés à manipuler le logiciel. Un apprentissage important s'est réalisé au niveau de la motricité et au niveau de l'orientation dans l'espace. Un travail sur le plan cognitif (structuration de l'espace) pourrait être entrepris pour arriver à une meilleure maîtrise du logiciel avant d'envisager d'autres utilisations.

Pour Fabrice (6 ans), le problème est le même que pour Aurélie : il a eu du mal à intégrer les règles d'utilisation. Cependant, OrdiThéâtre s'est révélé accessible pour lui et un travail d'apprentissage a été réalisé. Comme pour Aurélie, un travail est possible sur le plan cognitif (structuration de l'espace) et aussi sur le plan de la créativité.

Marie-Laure (5ans) a également eu difficile au début d'une part, à comprendre les commandes et d'autre part, à les manipuler en raison de son handicap. Cependant, cela ne l'a pas empêchée d'apprécier le logiciel et d'avoir une activité très riche à chaque séance. Elle s'est fortement identifiée à la petite fille et a exprimé, à travers elle, ses difficultés d'enfant handicapée (impuissance motrice et angoisse). OrdiThéâtre a permis une observation particulièrement intéressant de Marie-Laure et a fortement suscité sa projection dans les marionnettes et l'expression de ses problèmes. Pour elle, une utilisation psychothérapeutique semble particulièrement intéressante.

Nous avons ainsi exposé les commentaires des moniteurs, à propos du logiciel, en fonction des différents enfants. Il en ressort

qu'OrdiThéâtre est accessible à tous les degrés de handicap moteur. Le logiciel offre, aux enfants à handicap extrême, une possibilité d'être actifs et autonomes dans le jeu. Une utilisation ludique paraît particulièrement appropriée à ce type d'enfants vu le nombre limité de jeux auxquels ils ont accès. Un travail peut aussi, grâce à OrdiThéâtre, être effectué au niveau de la structuration de l'espace. Le logiciel est spécialement puissant dans une utilisation projective. Il a permis, pour 2 enfants, l'expression de problèmes qui n'avaient pas été détectés dans les situations habituelles et même dans des situations favorables à leur mise-à-jour (dessins, jeux,...). OrdiThéâtre fournit donc des possibilités diagnostiques et thérapeutiques. Il est à souligner que la puissance du logiciel constitue parfois une contre-indication pour certains enfants, comme pour Franck par exemple. Sa projection dans les marionnettes l'amène à un état de forte angoisse et ceci pourrait provoquer une sorte de bloquage psychologique. Une grande question se pose alors : peut-on laisser les enfants jouer seuls avec OrdiThéâtre ou est-il préférable de les accompagner constamment d'un adulte ? Nous ne pouvons répondre nous-même à cette question vu nos connaissances limitée en psychologie.

Mme de Barbot, psychologue, nous a exposé son avis en ce qui concerne l'utilisation d'OrdiThéâtre en fonction des âges des enfants :

Pour les enfants de 5 et 6 ans, OrdiThéâtre nécessite un apprentissage parfois important mais il est accessible à cette population.

Dans le cas d'enfants entre 10 et 11 ans, l'intérêt pour le logiciel s'épuise vite, une fois que l'enfant maîtrise toutes ses possibilités.

OrdiThéâtre est particulièrement bien adapté aux enfants entre 6 et 10 ans. Pour cette tranche d'âge, les utilisations sont multiples.

Les différentes utilisations d'OrdiThéâtre dépendent donc essentiellement de l'enfant avec lequel on travaille. L'âge, le caractère et les éventuels problèmes psychologiques de l'enfant déterminent s'il est préférable d'en dégager une utilisation ludique, pédagogique ou thérapeutique ou parfois même s'il existe une contre-indication à son utilisation.

OrdiThéâtre a également été placé, durant un mois, dans une école de l'enseignement spécial à Lessines. L'institutrice, Mme R. Deroose, étant seule avec les enfants, elle n'a pu travailler séparément avec chacun d'entre eux et remplir les questionnaires. Cependant, l'utilisation du logiciel en groupe est également intéressante. Les objectifs étaient essentiellement pédagogiques et ludiques. Les commentaires exprimés par cette institutrice furent très favorables. La version du logiciel comprenait la musique et un nombre plus élevé de décors et de marionnettes. Nous avons ainsi pu remarquer que les enfants étaient fortement attirés par la musique. En travaillant tous ensemble, ils ont découvert et appris le logiciel et ont créé des scènes. Il y a eu beaucoup d'interaction entre les enfants pour décider de ce qu'il fallait faire. Même les enfants les plus taiseux ont exprimé leur avis. Tous ont émis le désir de jouer et de manipuler OrdiThéâtre. Celui-ci a donc fortement favorisé l'expression et a permis une utilisation pédagogique intéressante.

Cette première évaluation du logiciel est donc très favorable et tend à montrer qu'OrdiThéâtre satisfait aussi bien les enfants que les adultes travaillant avec eux. Il atteint ses objectifs de départ en étant accessible à un maximum d'enfants et en permettant les différentes utilisations espérées par les moniteurs : ludiques, pédagogiques et thérapeutiques.

5. Le choix du matériel.

Le choix de l'ordinateur sur lequel développer une telle application est primordial. Il doit faire l'objet d'une réflexion sérieuse et est bien souvent le fruit d'un compromis entre diverses exigences contradictoires. Nous allons donc expliquer les différentes raisons qui nous ont poussé à porter notre choix sur l'Apple Macintosh, et nous décrirons ensuite de manière plus détaillée les caractéristiques importantes de cette machine.

5.1. Pourquoi le Macintosh?

Lorsque l'on développe un logiciel pour des enfants IMC, un des aspects les plus importants à prendre en compte est celui du coût du matériel. En effet, les institutions s'occupant de ces enfants n'ont bien souvent que peu d'argent à dépenser pour l'achat d'ordinateurs, et une application parfaite nécessitant une machine trop chère n'aurait aucune chance de succès. Une interface simple et conviviale au niveau du système est également recommandée, afin de faciliter l'accès aux différents programmes et afin de ne pas rebuter les utilisateurs potentiels. Enfin, les qualités ergonomiques et la possibilité de connecter, de préférence de manière transparente, divers matériels adaptés aux handicaps moteurs (par exemple un casque ultrasonique à la place de la souris) sont deux facteurs primordiaux dans le choix d'un ordinateur destiné à supporter une application pour enfants IMC.

Outre ces contraintes liées à la population visée, la nature même du programme que nous avions à développer exige certaines caractéristiques techniques. Une qualité graphique supérieure et de bonnes capacités sonores sont indispensables pour rendre attrayant un jeu de marionnettes sur ordinateur, mais également pour développer une interface conviviale, simple et adaptable à des enfants IMC. De plus, l'utilisation d'animation graphique nécessite un processeur performant et des capacités mémoire relativement importantes pour permettre des mouvements naturels et suffisamment rapides des personnages.

Enfin, réaliser un programme dans le cadre d'un mémoire suppose des limitations en temps et en moyens. Nous devions donc opter pour un ordinateur disponible aux facultés et offrant des outils de développement puissants, tant au niveau de la programmation que de la création de dessins ou de musiques.

Avant de partir en stage, nous avons envisagé plusieurs ordinateurs en fonction de ces différentes contraintes. A ce stade, trois types de machines se sont révélés utilisables:

- le Commodore Amiga, dont le principal avantage est d'exister en deux versions compatibles: l'Amiga 500, bon marché, pouvant

servir de machine d'exploitation et l'Amiga 2000, nettement plus puissant mais beaucoup plus cher, pouvant être utilisé comme machine de développement. Ces deux appareils offrent une très bonne résolution graphique en couleur ainsi que d'excellentes capacités sonores.

- l'Apple IIGS, peu coûteux (son prix est comparable à celui de l'Amiga 500) est assez répandu dans les institutions. L'écran couleur présente une résolution moyenne et les possibilités musicales sont bonnes, mais il s'agit d'une machine relativement limitée au niveau des facilités de développement.

- l'Apple Macintosh est généralement très apprécié des utilisateurs et des développeurs. Sa définition graphique est excellente et il offre une très bonne qualité au niveau du son. Il est cependant pénalisé par un prix relativement élevé en comparaison de l'Amiga 500 ou de l'Apple IIGS, ainsi que par un écran noir et blanc de taille réduite. La gamme des Macintosh II permet d'utiliser des écrans couleurs de grande dimension, mais un tel matériel est beaucoup trop cher pour une institution s'occupant de handicapés.

Pour fixer notre choix définitif, notre stage fut déterminant. Après une discussion avec Madame C. Charrière, ergothérapeute au service IMC de l'hôpital de Kremlin-Bicêtre et responsable du parc informatique de l'école, nous avons opté pour du matériel Apple. Tout d'abord, parce que l'établissement est exclusivement équipé en Apple (IIc, IIe, IIGS et Macintosh) et bénéficie d'avantages importants auprès de cette firme, comme par exemple une bonne assistance technique et des prêts ou des dons de matériel. Ensuite, parce que de très nombreux matériels spécifiques ont été développés afin d'améliorer l'accès aux ordinateurs Apple, ce qui rend ces derniers particulièrement bien adaptés aux IMC. Il nous restait donc à trancher en faveur de l'une des deux machines restant en compétition: le Macintosh et le IIGS.

L'Apple IIGS possède certains avantages non négligeables par rapport au Macintosh: son écran est plus grand, il est en couleur et son prix est moins élevé. Cependant, les quelques essais que nous avons

effectués avec cet ordinateur nous ont laissé un sentiment d'insatisfaction, tant au niveau de la fiabilité que des performances, de la qualité graphique ou des outils de développement. En conséquence, les restrictions techniques qu'imposaient l'utilisation de cette machine, étant donné le temps qui nous était imparti pour la réalisation du logiciel, nous semblaient un risque de trop diminuer les qualités du programme et donc de le rendre d'un moindre intérêt.

Le Macintosh représente un investissement plus important, mais qui reste raisonnable si l'on se cantonne dans les niveaux inférieurs de la gamme. Actuellement, les Macintosh les moins chers sont les modèles Plus et SE, dont les prix accusent une baisse certaine depuis quelques temps. Ces deux ordinateurs sont simples d'emploi, fiables et suffisamment puissants, mais ils ne disposent que d'un écran noir et blanc taille assez réduite (23 centimètres de diagonale). Cependant, l'excellente résolution graphique permet d'obtenir des images de très bonne qualité, précises et très attrayantes malgré l'absence de couleurs. A cela, il faut encore ajouter une interface conviviale et standardisée, une forte capacité mémoire, un très bon environnement de développement et de programmation, ainsi que l'existence d'une documentation complète et bien conçue. Il existe également une profusion de logiciels de haute qualité couvrant de très nombreux domaines, de l'éducation à la gestion en passant les jeux et les traitements de textes. Il s'agit donc de machines "bonnes à tout faire", dont le coût peut être justifié par l'étendue des services qu'elles peuvent rendre. C'est donc pour ces différentes raisons, ajoutées à la certitude d'un avenir prometteur pour une gamme qui constitue le cheval de bataille d'Apple, que nous avons décidé de développer OrdiThéâtre sur Macintosh. La configuration minimale nécessaire à l'utilisation du logiciel est constituée d'un Macintosh Plus, qui est le moins cher de la gamme, équipé d'un deuxième lecteur de disquettes.

5.2. Quelques caractéristiques importantes du Macintosh.

Depuis l'introduction en 1984 du premier Macintosh 128K, la gamme s'est largement étendue et a fortement gagné en puissance. Tous

les représentants de la famille Macintosh sont compatibles entre eux, offrent la même qualité graphique et proposent la même interface conviviale. Ils se distinguent notamment par leur aspect (*ligne compacte* avec écran 23 centimètres incorporé, *ligne modulaire* avec écran séparé ou *ligne portable*), leur capacité mémoire maximale, les cartes vidéos qu'ils peuvent recevoir (certaines de ces cartes peuvent supporter des écrans de taille A3 ou proposent plusieurs millions de couleurs). Les processeurs utilisés sont également différents d'un modèle à l'autre: les Macintosh Plus et SE se contentent du Motorola MC68000 cadencé à 8 mégahertz (MHz) alors que les autres utilisent le Motorola MC68030 cadencé à 16 MHz (Mac SE/30, IIx et IIcx), 25 MHz (Mac IIci) ou 40 MHz (Mac IIfx). Nous décrirons ici plus particulièrement les caractéristiques des Macintosh Plus et SE, les moins chers de la gamme, mais beaucoup de considérations sont également valables pour le reste de la "famille".

5.2.1. Quelques caractéristiques hardware.

Le processeur utilisé par les Macintosh Plus et SE est donc le Motorola MC68000 tournant à 8 MHz. Il s'agit d'un processeur 16 bits, comportant seize registres universels - c'est à dire pouvant être utilisés pour n'importe quelle opération - de 32 bits chacun. L'espace d'adressage est linéaire, ce qui signifie que la mémoire est traitée comme une chaîne continue d'adresses, sans segmentation. Le bus des adresses ayant une taille de 24 bits, l'espace adressable maximal est de 16 mégaoctets (ou MB, pour megabyte). Afin de contrôler les différentes unités d'entrée/sortie, le processeur est épaulé par un contrôleur de communications sérielles, un adaptateur souple d'interface (Versatile Interface Adaptator, VIA) et une unité de commande de disques [Duff 1985].

La mémoire RAM (Random Access Memory, ou mémoire vive) est, sur les deux machines, de 1 MB et est extensible à 4 MB. En plus de la mémoire vive, le Macintosh Plus dispose d'une ROM (Read Only Memory, ou mémoire morte) de 128 kilobytes (KB) et le Macintosh SE d'une ROM de 256 KB, contenant les différentes fonctions de la boîte à

outils (voir infra). Au niveau des mémoires de masse, le Macintosh Plus se contente d'un seul lecteur de disquettes 3,5'' de 800 KB, alors que le Mac SE est équipé soit de deux lecteurs de disquettes 3,5'' de 1,4 MB, soit d'un seul lecteur 1,4 MB et d'un disque dur interne de 20 ou 40 MB. A ceci, il faut ajouter que tant le Macintosh Plus que le Macintosh SE disposent d'un port rapide SCSI (Small Computer Standard Interface) permettant de connecter en chaîne des lecteurs de disquettes ou des disques durs externes supplémentaires.

La résolution graphique d'un Macintosh est de 72 points par pouces, tant horizontalement que verticalement. Les modèles Plus et SE disposent d'un écran de 23 centimètres de diagonale, affichant 512 * 342 pixels. Comme il s'agit d'un système noir et blanc, chaque point de l'écran n'a que deux états possibles - il est soit allumé, soit éteint - ce qui permet de stoquer un pixel sur un seul bit. En conséquence, la mémoire d'affichage du Macintosh n'occupe que 512 * 342 bits, soit 21888 octets de mémoire RAM. Ceci permet de faire de l'animation graphique en transférant rapidement des écrans complets vers la mémoire d'affichage sans consommer trop de place en RAM.

Les Macintosh comportent une partie de synthèse sonore qui produit jusqu'à quatre voix simultanées, capable de synthétiser ou de reproduire des ondes aussi complexes que celles de la parole humaine. Cette section consomme, lorsque les quatres voix sont utilisées simultanément, au maximum 50 pourcents du temps du processeur MC68000, ce qui permet la poursuite des autres traitements pendant la production du son [Duff 1985]. Elle permet aussi bien d'utiliser des musiques digitalisées que de programmer ses propres sons. Il existe même des possibilités de synthèse vocale à partir d'un texte à prononcer et ce, grâce à un driver particulier qu'il suffit de copier sur le disque contenant le système. Malheureusement, ce driver n'a encore été développé qu'en version anglaise. Les Macintosh Plus et SE ne comportent qu'un seul haut-parleur interne, mais une prise externe permet d'utiliser leurs capacités stéréophoniques via un simple amplificateur audio. Pour terminer, nous pouvons encore signaler la possibilité d'ajouter une interface MIDI destinée à augmenter les possibilités musicales du Macintosh.

5.2.2 Les fichiers.

L'organisation des fichiers est particulièrement originale et intéressante. En effet, un fichier Macintosh est constitué de deux parties, ou *fourches*. La première partie, appelée *Data Fork* (fourche des données), correspond à la notion classique de fichier de données: elle contient des données quelconques représentées sous forme d'une suite d'octets et n'ayant théoriquement de signification que pour le programme qui a créé le fichier. On peut l'utiliser soit en accès séquentiel, soit en accès direct. La deuxième partie, appelée *Resource Fork* (fourche des ressources), contient toutes les ressources nécessaires au système ou à une application. Ces ressources peuvent par exemple être des textes, des dessins, des sons, des types de caractères, des menus, des spécifications de dialogues, des segments de code de programme ou tout autre chose jugée nécessaire au fonctionnement d'un logiciel. Elle sont structurées par types et peuvent être manipulées par n'importe quelle application. Lorsqu'un programme ouvre la partie des ressources d'un fichier, le système maintient automatiquement en mémoire une table d'index qui assure un accès rapide aux éléments désirés. L'utilisation de ressources permet notamment de séparer les textes des menus et des dialogues du reste du programme, ce qui facilite grandement la traduction des logiciels. Elle autorise également la personnalisation du software par les utilisateurs qui peuvent facilement changer les dessins, les sons, les textes et les icônes des applications qu'ils ont achetées.

5.2.3. L'interface conviviale.

Une clé importante du succès du Macintosh est son interface graphique conviviale, standardisée, facile à utiliser et à apprendre pour des non informaticiens. Elle tire profit de la haute résolution graphique et se base sur des notions assez simples et intuitives.

Des fenêtres sont utilisées pour présenter les informations et les regrouper de manière logique. Elles peuvent être déplacées, agrandies ou

rétrécies à volonté par l'utilisateur qui peut ainsi organiser l'écran comme bon lui semble. Les dialogues prennent également place au sein de fenêtres et ils ne nécessitent que très peu d'entrées au clavier grâce à l'utilisation de boutons simples, de boutons radios, de checkboxes et de cadrans.

La manipulation directe permet d'effectuer des actions sur des fichiers ou des objets virtuels au moyen de la souris. On peut par exemple lancer un programme, copier ou effacer un fichier sans utiliser le clavier et sans passer par des menus. Les icônes utilisées pour figurer les différents objets sont assez claires et compréhensibles: une poubelle permet de jeter des documents, qui sont eux-mêmes représentés par une feuille de papier.

Des menus déroulants sont manipulés à l'aide de la souris. Bien présentés et n'apparaissant que lorsque l'on clique dans la barre de menus, ils sont assez faciles d'emploi et évitent l'apprentissage de commandes complexes. Ils peuvent également être activés par le clavier, en utilisant des raccourcis.

L'ensemble des règles de bonne conduite édictées par Apple en matière d'interface homme-machine sont publiées dans un livre appelé "Human Interface Guidelines: The Apple Desktop Interface". Ce livre est censé servir de base à toute personne développant des applications sur Macintosh.

5.2.4. La boîte à outils.

Deux choses sont assez frappantes lorsque l'on utilise un Macintosh: pratiquement tous les programmes utilisent la même interface homme-machine standardisée, et de nombreuses applications offrent une compatibilité étonnante, permettant par exemple d'intégrer dans un traitement de texte des dessins faits avec un logiciel graphique. Ceci n'est pas uniquement dû aux recommandations d'Apple qui abondent dans ce sens, mais plutôt à la fantastique *Toolbox* ou boîte à outils. Celle-ci regroupe dans la ROM du Macintosh plus de 900 fonctions permettant d'écrire facilement des programmes utilisant des menus déroulants, des

fenêtres superposées, des dialogues, des barres de défilement et tout ce qui fait qu'un programme ressemble à une application pour un Macintosh. Nous allons brièvement voir comment fonctionne la boîte à outils et quels en sont les composants principaux.

Les fonctions de la boîte à outils peuvent être appelées depuis n'importe quel langage, généralement via une interface propre au compilateur. Ceci permet au programmeur d'accéder aux routines se trouvant en ROM de manière transparente, en utilisant les conventions d'appel du langage utilisé. Au niveau de la machine, tous les appels aux fonctions de la Toolbox doivent être transformés en sauts vers l'adresse en ROM de la première instruction de la routine désirée. Pour ce faire, le système du Macintosh utilise une particularité du processeur Motorola, à savoir l'*emulator trap*. Une *emulator trap* arrive lorsque le processeur rencontre une instruction qu'il ne connaît pas. Elle provoque l'arrêt momentané de l'exécution normale, le sauvetage des registres et le démarrage d'une routine de traitement d'exception qui, dans le cas du Macintosh, va chercher l'adresse de la routine voulue et lancer son exécution. Il en résulte que la boîte à outils devient une extension logique de l'ensemble d'instructions du processeur [Duff 85].

Le contenu de la boîte à outils est divisé en trois parties [Chernicoff 1987]:

- le système d'exploitation gère les tâches de bas niveau comme la gestion de la mémoire, les entrées-sorties disques et les communications sérielles.

- les routines graphiques QuickDraw sont responsables de tout ce qui est affiché à l'écran. Elles sont à la fois puissantes et très rapides, elles sont capables de manipuler aussi bien des bitmaps que des régions, des textes ou des formes de toutes sortes (rectangles, cercles, polygones,...).

- la User Interface Toolbox implémente les constructions de haut niveaux comme les fenêtre et les menus. Elle contient notamment des routines de gestion de fenêtres, de dialogues, de menus, de fichiers, de ressources, de sons ou d'édition de textes.

Cette rapide description technique est bien entendu incomplète, les lecteurs intéressés par les caractéristiques du Macintosh trouveront de plus amples renseignements dans les différents livres mentionnés dans la bibliographie.

6. Architecture et spécifications du programme.

Les phases de conception de l'architecture et de spécification d'un logiciel sont deux étapes très importantes du développement. Faute de place, nous ne pouvons nous permettre d'inclure ici l'entièreté de ces spécifications. Nous nous contenterons donc de rappeler quelques principes de base du développement d'une architecture logicielle avant de décrire et de spécifier l'architecture logique et de décrire l'architecture physique. Enfin, nous terminerons cette section par quelques remarques sur l'environnement de programmation, ainsi que sur les techniques d'animations graphique et musicale que nous avons utilisées.

6.1. Principes de développement d'une architecture logicielle.

Concevoir une architecture logicielle revient à construire une structure pour un système à développer. Cette structure comprend divers composants entre lesquels existent des relations logicielles, comme par exemple les relations "appelle" ou "utilise", devant être bien définies. Pour assurer les qualités du système (robustesse, fiabilité, portabilité, convivialité, facilité de maintenance,...) tout en obéissant aux contraintes organisationnelles (confidentialité, par exemple) et aux contraintes liées à l'environnement, une bonne architecture doit de préférence respecter deux principes de bases: la hiérarchisation et la modularité [van Lamsweerde 1988].

La *structuration hiérarchique* permet de découper le système en différentes "couches", ou niveaux d'abstraction. Ces niveaux vont être organisés hiérarchiquement, sur base d'une relation logicielle particulière. De manière formelle, nous dirons qu'une structure est

hiérarchisée si et seulement si il existe une relation R entre les composants telle que (voir figure 4.6.1):

1. niveau 0 = {A : ∄ B | R (B,A) }
2. niveau i = {A : ∃ B ∈ niveau i-1 | R (B,A)
   & R (C,A) ⇏ C ∈ niveau ≤ i-1}

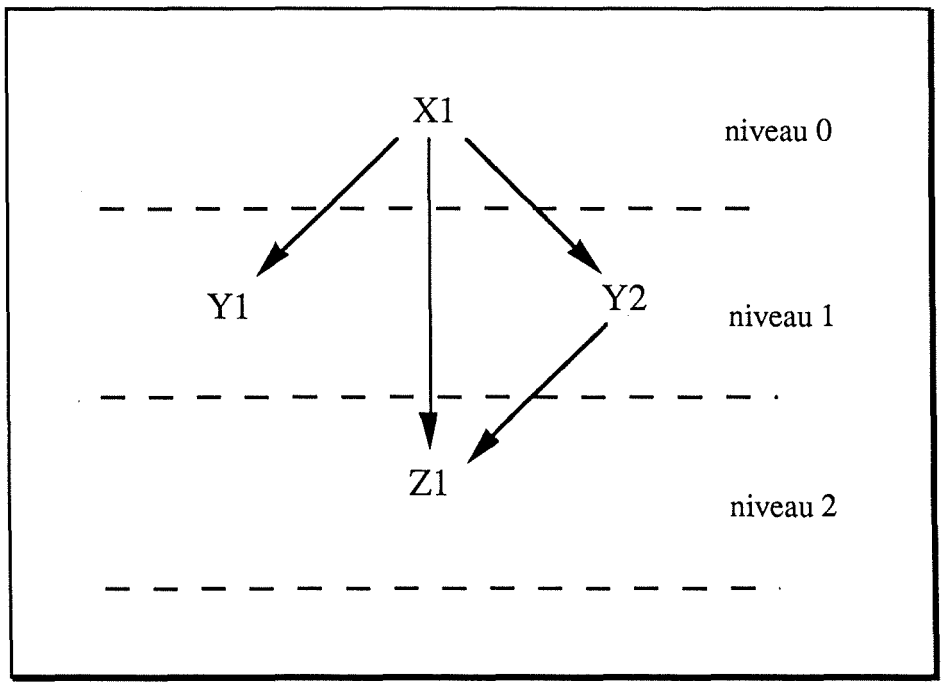


Figure 4.6.1.: exemple de hiérarchie.

Nous dirons qu'une structure est *modulaire* si les composants qui la forment et les relations entre ces composants sont tels que:

1. leurs attributs (spécifications, modes de réalisation, procédés de construction et performances) peuvent être définis de manière simple et précise.

2. chaque composant de la structure possède trois qualités:

- il présente une forte capacité à cacher de l'information.

- il présente un faible degré de couplage avec les autres composants.

- il présente une forte cohérence interne.

Les objectifs d'une bonne modélisation sont le fractionnement du travail (travail de documentation, de conception et de programmation par exemple), la répartition effective de celui-ci (développement en parallèle et autonomie des personnes) et la réduction des coûts de maintenance grâce à la division du système en petites parties autonomes. Une bonne découpe modulaire permettra également de concevoir l'interface homme-machine séparément du reste du programme, assurant ainsi son indépendance vis-à-vis de la tâche.

Deux niveaux d'architecture sont généralement distingués:

1. l'architecture logique, dont les composants sont des unités de travail pour l'analyste, comme par exemple des unités de spécification, de documentation ou de modification. Les relations mises en œuvre à ce niveau sont notamment les relations "utilise", "instancie", "importe/exporte" ou "hérite".

2. l'architecture physique, dont les composants sont des unités de travail pour la machine, comme par exemple des unités de compilation, d'exécution ou de texte. Les relations entre les composants seront notamment les relations "appelle", "déclenche" ou "active".

6.2. L'architecture logique.

Pour structurer hiérarchiquement notre architecture logique, nous nous sommes basés sur la relation "utilise", définie comme suit: "un composant A utilise un composant B si et seulement si la validité de A dépend de la disponibilité d'une version correcte de B". Nous commencerons par une description de l'architecture, puis nous présenterons les spécifications externes des différents modules qui la composent.

6.2.1. Description de l'architecture logique.

La représentation graphique de l'architecture (figure 4.6.1) nous montre une structure hiérarchique comprenant huit modules regroupés en cinq niveaux.

Le niveau le plus élevé ne contient qu'un module: *OrdiThéâtre*. Ce dernier est ce que l'on appelle généralement le coordinateur: il est le dépositaire du scénario d'exploitation, lui seul organise les séquences d'actions lors d'une séance d'utilisation. Si l'on voulait changer le scénario, seul ce module devrait être modifié.

Le niveau suivant constitue le niveau fonctionnel: il regroupe les différentes tâches dégagées lors de l'analyse fonctionnelle. Etant donné leur nombre peu élevé, leur similitude et leur forte cohésion, nous avons choisi de placer ces fonctions dans un seul module, appelé *créativité*. Celui-ci propose donc les fonctionnalités suivantes: choisir un décor, choisir des personnages, choisir une musique, choisir une scène à regarder, visualiser une scène et créer une scène. Une autre approche aurait par exemple pu conduire à grouper les quatre premières fonctions dans un module *choix* et les deux dernières dans un module *création-visualisation*. Cependant, nous avons préféré tout mettre dans une seule "boîte" en raison de la forte liaison qui existe entre les choix et la création ou la visualisation de scènes.

Le troisième niveau contient également un seul module, destiné à prendre en charge l'interface homme-machine. Il permet l'indépendance des tâches à réaliser vis-à-vis des modes de dialogue et d'interaction avec l'ordinateur. La présence de ce composant dans l'architecture facilite également la conception d'interfaces différentes adaptées à divers handicaps.

L'avant-dernier niveau se compose de quatre modules regroupant des outils réutilisables dans un contexte différent, à savoir:

- le module *musique* dont la fonction est de jouer des thèmes musicaux en processus de fond, c'est à dire sans stopper l'exécution d'autres tâches.

- le module *animation* permet de gérer l'animation graphique de personnages se déplaçant dans des décors.

- le module *enregistrement* enregistre et lit des scènes et des traces.

- le module *menus* regroupe des utilitaires de gestion de menus de types différents.

Ces quatres composants constituent une boîte à outils de haut niveau, dont l'utilisation facilite grandement l'implémentation des modules plus élevés dans la hiérarchie.

Enfin, le dernier niveau contient un module de gestion de la B.D. (base de données). En fait, cette dernière est constituée d'une bibliothèque de dessins à laquelle viennent s'ajouter des scènes, des traces et des paramètres. Ce module permet de masquer aux niveaux supérieurs l'organisation physique de la B.D. et favorise ainsi la portabilité du logiciel.

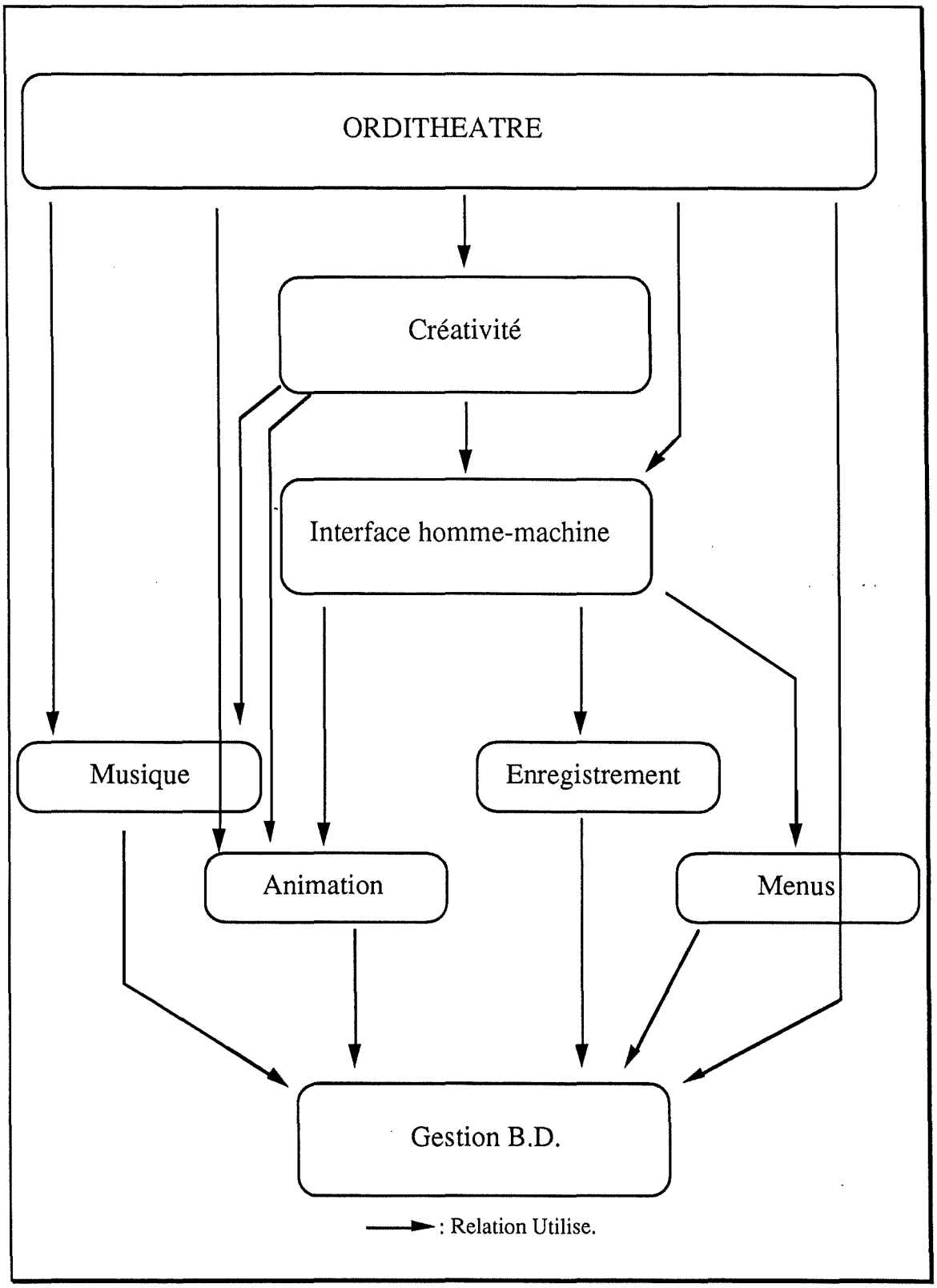


Figure 4.6.2.: architecture logique.

6.2.2. Spécifications externes des modules.

Nous allons maintenant décrire de manière plus précise les spécifications externes des principaux composants de l'architecture logique. Ces spécifications sont présentées sous forme d'assertions en français présentant les arguments nécessaires (Arg.), les préconditions que ces arguments doivent satisfaire (Pré.) et les postconditions qu'ils présentent en sortie du module (Post.).

* Module créativité.

Arg.: la fonction à réaliser, la B.D., **décor**, **musique**, **personnages**, **scène**.

Pré.: la fonction est "choisir une scène", "choisir un décor", "choisir une musique", "choisir des personnages", "créer une scène" ou "visualiser une scène".

Post.:- si la fonction est "choisir une scène", "choisir un décor", "choisir une musique" ou "choisir des personnages", la structure **scène**, **décor**, **musique** ou **personnages** identifie la scène enregistrée, le décor, la musique ou les personnages qui a (ont) été choisi(s).

- si la fonction est "créer une scène": une scènette a été jouée par l'utilisateur avec le décor, le thème musical et les personnages spécifiés par **décor**, **musique** et **personnages**. Si l'utilisateur a décidé d'enregistrer la scène, la B.D. est mise à jour.

- si la fonction est "regarder une scène": si une scène enregistrée était spécifiée par **scène** avant l'activation de la fonction, elle s'est déroulée à l'écran jusqu'à ce que l'utilisateur ait décidé de stopper la visualisation. Si l'utilisateur a demandé à recréer la scène, la fonction "créer une scène" a été activée.

* Module interface homme-machine.

Arg.: les paramètres relatifs à l'utilisateur, la B.D., **lire-scène**, **enregistrer-trace**, **commande**.

Pré.: les paramètres spécifient un mode de dialogue valide.

Post.:- si **lire-scène** n'identifie pas une scène enregistrée, **commande** contient une action à effectuer demandée par l'utilisateur via le mode de dialogue spécifié par les paramètres. Sinon, **commande** contient soit une action demandée par l'utilisateur via le mode de dialogue spécifié par les paramètres, soit une action lue dans la scène enregistrée spécifiée par **lire-scène**.

- si **enregistrer-trace** identifie une structure de trace, la B.D. est mise à jour afin de conserver la trace des manipulations de l'utilisateur.

* Module musique.

Arg.: la B.D., une fonction à réaliser, **musique**.

Pré.: la fonction est "charger une musique", "jouer la musique" ou "arrêter la musique".

Post.:- si la fonction est "charger une musique" et que **musique** identifie un thème musical, alors ce thème est prêt à être joué.

- si la fonction est "jouer la musique" et qu'un thème musical est prêt à être joué, alors ce dernier se joue de manière continue en processus de fond.

- si la fonction est "arrêter la musique", le thème musical est stoppé.

* Module animation.

Arg.: une fonction à effectuer, la B.D., **décor**, **personnage**.

Pré.:- soit la fonction est "charger un personnage" ou une fonction d'animation ("déplacer à gauche", "déplacer à droite", "monter", "descendre", "lever", "asseoir", "coucher", "pivoter à gauche", "pivoter à droite", "positionner", "entrer en scène", "sortir de scène") et **personnage** spécifie un objet qui peut être animé,

- soit la fonction est "placer un décor" et **décor** spécifie une image.

Post.:- si la fonction est "charger un personnage": l'objet spécifié par **personnage** est prêt à être animé.

- si la fonction est une des fonctions d'animation et si l'objet spécifié par **personnage** est prêt à être animé, alors ce dernier effectue à l'écran l'action demandée.

- si la fonction est "placer un décor", l'image spécifiée par **décor** est affichée à l'écran et servira de toile de fond pour toutes les fonctions d'animation.

* Module enregistrement.

Arg.: la B.D., une fonction à réaliser, **commande**, **trace-ou-scène**.

Pré.:- soit la fonction à réaliser est "enregistrer dans une scène", **trace-ou-scène** identifie une scène enregistrée et **commande** contient une fonction du module animation,

- soit la fonction à réaliser est "lire dans une scène" et **trace-ou-scène** identifie une scène enregistrée,

- soit la fonction est "enregistrer dans une trace", **trace-ou-scène** identifie une structure de trace et **commande** représente une action de l'utilisateur.

Post.:- si la fonction est "enregistrer dans une scène": **commande** est ajoutée à la fin de la scène spécifiée par **trace-ou-scène**, la B.D. est mise à jour.

- si la fonction est "lire dans une scène": **commande** contient une fonction du module animation provenant de la scène enregistrée spécifiée par **trace-ou-scène**.

- si la fonction est "enregistrer dans la trace", **commande** est ajoutée à la fin de la trace spécifiée par **trace-ou-scène**, la B.D. est mise à jour.

* Module menus.

Remarque: pour plus de simplicité, nous appellerons "menus" les menus déroulants destinés au paramétrage du logiciel par le moniteur, et "lignes de commandes" les menus permettant à l'enfant de communiquer avec le programme.

Arg.: une fonction à réaliser, **commande**.

Pré.: la fonction à réaliser est "afficher la barre de menus", "effacer la barre de menus", "suivre la souris dans les menus", "placer une marque dans un menu", "retirer une marque dans un menu", "définir une ligne de commandes", "positionner le curseur dans la ligne de commandes", "supprimer le curseur de la ligne de commandes", "placer une marque dans la ligne de commandes" ou "effacer une marque dans la ligne de commandes".

Post.: la fonction est effectuée. Si celle-ci est "suivre la souris dans les menus", **commande** contient l'item qui a été choisi dans les menus déroulants.

6.3. L'architecture physique.

Une fois l'architecture logique spécifiée et avant de passer au codage des modules, il faut choisir une identité physique pour les composants et les relations. Ceci se fait en définissant une architecture physique, qui doit de préférence essayer de réunir deux qualités: la traçabilité (c'est-à-dire qu'elle doit refléter le mieux possible l'architecture logique) et la performance. Nous allons d'abord décrire les notions sur lesquelles nous avons basé cette architecture avant d'en faire une description succincte. Les spécifications des modules sont incluses en commentaires dans le texte du programme, fourni en annexe.

6.3.1. Type des composants et relations.

Nous avons indiqué précédemment que les composants de l'architecture physique sont des unités de travail pour la machine, par exemple des unités de compilation. Comme nous avons développé le programme en Pascal, nous avons tout naturellement choisi d'appuyer l'architecture physique sur la notion de *unit.*. Au sens Pascal du terme, une unit est une collection de constantes, de types, de variables, de procédures et de fonctions. Elle est compilable séparément et comporte deux parties distinctes: la section *interface,* qui contient les déclarations des constantes, types, variables, procédures et fonctions accessibles aux utilisateurs de la unit, et la section *implémentation,* qui contient le code implémentant les routines et dont le contenu est strictement privé. Il est à remarquer que l'environnement de programmation utilisé, à savoir le Lightspeed Pascal développé par THINK Technologies, encourage la modularité en facilitant grandement l'utilisation des units (voir infra).

La relation sur laquelle nous avons basé la hiérarchie est la relation "appelle", qui correspond à la notion, quelque peu étendue, d'appel de fonctions et procédures en Pascal. Nous dirons donc qu'un module A appelle un module B si et seulement si le module A fait appel à au moins une fonction ou procédure du module B, ou s'il fait référence à au moins une constante, une variable ou un type déclaré(e) dans le module B.

6.3.2. Description de l'architecture physique.

Le graphe de l'architecture physique nous montre une structure hiérarchique comprenant onze modules regroupés en six niveaux. Nous allons décrire ces modules en indiquant leurs correspondants au niveau logique.

Le module logique *OrdiThéâtre* est implémenté par le module physique portant le même nom et contenant le programme principal (au sens Pascal).

Au module logique *créativité* correspond également un seul module de même nom. Il contient sept procédures et fonctions implémentant les sept fonctionnalités décrites dans l'architecture logique.

Pour faciliter la programmation et la maintenance, le module logique *interface homme-machine* est implémenté par trois modules physiques distincts:

- le module *écrans* comporte six routines destinées à produire les différents écrans présentés dans la description de l'interface.

- le module *gestion-dialogues* est composé de dix routines prenant en charge la gestion des différentes fenêtres de dialogue (fenêtres de confirmation, de saisie du nom, de définition de paramètres...).

- le module *commandes* gère les entrées de l'utilisateur au moyen de trois fonctions principales correspondant aux trois modes d'entrée possibles (souris, clavier-flèches, clavier-défilement). Comme ces routines sont chargées de détecter et filtrer les actions de l'utilisateur, elles assurent aussi l'enregistrement des traces. Ce sont également elles qui vont lire les commandes provenant de scènes enregistrées et ce, de manière à pouvoir facilement donner la priorité aux entrées de l'utilisateur.

Le module *musique,* implémentant le module logique homonyme, regroupe six procédures et une variable d'état permettant une gestion simple et efficace des thèmes musicaux.

Le module physique *animation* est un des modules les plus complexes et, avec ceux réalisant l'interface, un des plus importants du programme. Un soin tout particulier a dû être apporté à sa réalisation pour obtenir des performances et une efficacité suffisantes. Outre des descriptions de constantes, types et variables implémentant les différents objets qu'il manipule, ce module contient 25 procédures destinées à réaliser toutes les possibilités d'animation du logiciel.

Le module physique *enregistrement* définit les structures de données représentant les traces et les scènes enregistrées et fournit onze routines destinées à les manipuler.

Pour plus de clarté, le module logique *menus* a été découpé en deux modules physiques:

- le module *lignes de commandes* contient huit routines et des structures de données permettant de gérer les menus destinés à l'enfant.

- le module *gestion-menus* regroupe huit routines destinées à gérer les menus déroulants destinés au moniteur.

Enfin, le module logique *gestion B.D.* est implémenté par le module logique *fichiers,* qui comporte les structures de données et 23 procédures et fonctions permettant de manipuler les fichiers contenant les images, les scènes enregistrées et les traces.

Comme on peut le remarquer en comparant les graphes des deux architectures, nous nous sommes efforcés de refléter au mieux l'architecture logique que nous avions développée. Seuls deux modules logiques (*interface homme-machine* et *menus*) ont été éclatés en plusieurs modules physiques, et ce, afin de faciliter l'implémentation et la maintenance. Nous avons également essayé de construire une architecture permettant de bonnes performances en évitant les fonctions par trop "monstrueuses" et les intermédiaires inutiles.

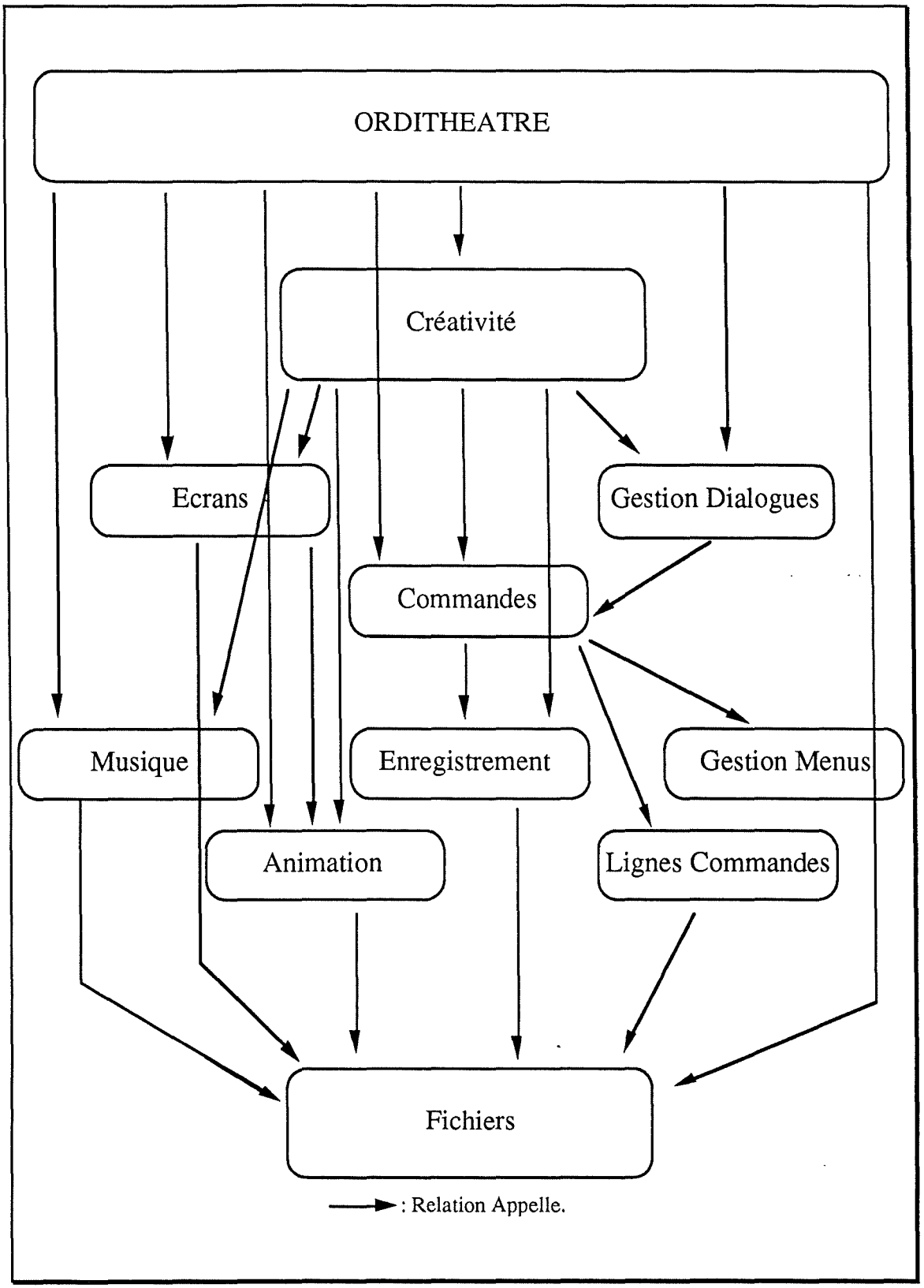


Figure 4.6.3.: architecture physique.

6.4. Remarques sur l'environnement de programmation.

Nous avons déjà indiqué que le programme a été développé sur Apple Macintosh, à l'aide de la version 2.0 du Lightspeed Pascal proposé par THINK Technologies. Nous présentons dans cette section quelques remarques générales sur le Macintosh d'abord et sur le Lightspeed Pascal (appelé aussi THINK Pascal) ensuite. Nous espérons ainsi faciliter la tâche de personnes voulant se lancer dans la programmation sur Macintosh, en essayant de leur permettre d'éviter les écueils auxquels nous nous sommes heurtés.

6.4.1. A propos du Macintosh.

Le Macintosh est une machine puissante offrant des facilités remarquables au programmeur habitué, mais qui peut se révéler un véritable casse-tête pour le débutant non averti. Il n'est en effet pas facile de s'y retrouver parmi les quelques 900 fonctions de la Toolbox, mais vouloir les ignorer et développer une application en se basant uniquement sur les fonctions standard d'un langage serait quelque peu "suicidaire". Non seulement ce serait une perte de temps, mais le programme ne pourrait atteindre la même qualité et le programmeur risquerait d'être confronté à des problèmes inextricables. Le premier conseil que nous pouvons donner est donc de s'entourer d'une documentation abondante et de la lire avec attention, afin de s'imprégner de la philosophie sous-jacente au Macintosh. Voici une liste, non exhaustive, de quelques livres qui nous semblent intéressants:

- **Programmer's Introduction to the Macintosh Family**, publié par Apple, est un excellent point de départ. Il ne s'agit pas d'une documentation technique, mais d'une très bonne introduction aux différents concepts sous-jacents au Macintosh. Il aide grandement à maîtriser les différents manuels de référence.

- les cinq volumes du **Macintosh Revealed** de Stephen Chernicoff contiennent des descriptions très précises et complètes des fonctions de la Toolbox, ainsi que des principes de

programmation à respecter. Très bien écrits, ils font comprendre facilement et avec beaucoup d'humour des notions parfois complexes et très techniques.

- **Inside Macintosh** (actuellement six volumes), publié par Apple, constitue la "référence ultime" pour développer des programmes sur Macintosh. Il est mis à jour par la publication de nouveaux tomes à chaque évolution conséquente du système et du matériel. D'un accès relativement complexe, nous conseillons de n'utiliser Inside Macintosh qu'en dernier recours, par exemple lorsque la référence recherchée ne se trouve pas dans le Macintosh Revealed.

- enfin, **Human Interface Guidelines: The Apple Desktop Interface**, également publié par Apple, est une référence quasiment indispensable aux personnes désirant commercialiser leurs programmes. En effet, les utilisateurs inconditionnels du Macintosh tiennent très fort à "leur" interface standard, et le respect de ses grandes lignes est une condition sine qua non pour un succès commercial.

Cette liste de livres est bien entendu largement incomplète, mais elle constitue une base solide pour développer sans trop de peine des applications de bonne qualité sur Macintosh.

Un deuxième conseil qui nous semble important est d'essayer de respecter les lignes de conduites édictées par Apple, tant au niveau de la "philosophie" (The User plays a central role), que de l'interface homme-machine, des règles de compatibilité ou des structures et techniques de programmation. En effet, la simplicité d'utilisation du Macintosh masque une grande complexité interne, et le non respect de ces règles pourrait provoquer des erreurs et des problèmes insoupçonnés.

Il nous semble également très important de faire un usage intensif des ressources: non seulement elles constituent une manière élégante, pratique et performante de stocker et récupérer certaines informations, mais en plus, elles facilitent l'évolution et l'adaptation des logiciels. Par exemple, dans OrdiThéâtre, tous les textes, dessins et icônes sont stockés sous forme de ressources. Ceci permet la modification aisée des icônes et

des textes ou l'ajout de personnages et décors par une personne non informaticienne, sans toucher au programme, en utilisant un éditeur de ressources relativement simple et puissant (ResEdit).

Enfin, si l'apprentissage de la programmation sur Macintosh peut sembler quelque peu fastidieuse lorsque l'on est habitué à un ordinateur de type PC, nous pensons que l'effort en vaut la peine, tant les facilités offertes par la boîte à outils sont remarquables. En résumé et en guise de conclusion, nous donnerons un dernier conseil aux personnes tentées par le développement d'applications pour cet ordinateur: "Think Macintosh!"

6.4.2. Le Lightspeed Pascal.

La version 2.0 du Lightspeed Pascal constitue un environnement de développement intégré très complet et agréable. Il favorise une programmation modulaire en facilitant l'édition, l'intégration, la compilation et le reliage (linkage) des nombreux fichiers composant un programme (un fichier contient généralement une unit, une librairie ou le programme principal) et ce, grâce à la notion de *projet.* Le projet est au cœur de l'environnement de développement Lightspeed Pascal [THINK Technologies 1988]. Il contient le code objet de tous les fichiers sources déjà compilés, et maintient les dépendances entre les morceaux de code. Il garde également la trace de tous les fichiers qui doivent être recompilés suite à une modification faite à l'aide de l'éditeur. Enfin, le projet garde également la trace des informations destinées au debugger de niveau source.

Les principaux outils intégrés dans cet environnement sont les suivants:

- un gestionnaire de projets qui permet de créer, modifier et manipuler l'ensemble du projet. Il maintient automatiquement toutes les informations nécessaires au compilateur, au linker, et au debugger.

- un éditeur syntaxique qui permet d'éditer les différents fichiers constituant le programme. Spécialement conçu pour le Pascal, il

formatte les textes et vérifie la syntaxe chaque fois que l'on tape un point-virgule ou un caractère de fin de ligne (<return>).

- un compilateur rapide et "intelligent" qui ne recompile que les modules nécessaires lorsque l'on fait des modifications.

- un linker ayant des capacités de "smart linking", c'est-à-dire capable d'éliminer du code objet final toutes les parties de code non utilisées (par exemple, des fonctions déclarées et implémentées mais jamais appelées).

- un debugger de niveau source permettant notamment l'exécution pas à pas, la trace, l'insertion de points d'arrêt dans le code source et la visualisation de variables. Il est également possible de stopper l'exécution du programme et revenir au debugger à n'importe quel moment. Il s'agit d'un outil puissant et très pratique pour la mise au point d'une application.

Le Lightspeed Pascal permet l'accès à l'ensemble des routines comprises dans la Toolbox, mais il implémente également toutes les fonctions du Pascal standard (à l'exception de la fonction *Assign:* sur un Macintosh, le nom physique du fichier est spécifié lors de l'appel à la fonction *Open).* Cependant, lorsqu'il existe une routine de la boîte à outils effectuant un travail comparable à celui d'une fonction ou procédure Pascal, il est généralement préférable d'utiliser la première citée. En effet, la fonction provenant de la Toolbox est bien souvent plus rapide, risque moins de provoquer des erreurs système et n'augmente pas la taille du code source, puisqu'elle se trouve en ROM.

A notre avis, le Lightspeed Pascal constitue un excellent environnement pour le développement de programmes assez importants (un projet peut contenir jusqu'à 255 fichiers différents). Cependant, lorsque le temps de développement est particulièrement critique, mais que les performances de l'application sont un souci secondaire, la solution Hypercard (un environnement de développement orienté objet conçu pour le Macintosh par Apple) pourrait être préférable.

6.5. L'animation graphique.

Bien que limitée à quelques mouvements simples, l'animation graphique est une caractéristique importante d'OrdiThéâtre. Elle est en effet un attrait essentiel pour les enfants et donne au jeu un aspect dynamique, plus amusant et plus naturel. Il nous fallait mettre au point une technique simple et efficace, permettant des mouvements non saccadés sans utiliser un trop grand nombre de dessins. Nous allons présenter, sans rentrer dans trop de détails, la solution à laquelle nous sommes arrivés après de nombreux essais.

6.5.1. Les principes de base.

Pour animer un objet dans un décor fixe, il est nécessaire de décomposer le mouvement à effectuer en plusieurs images différentes. Ces images peuvent alors être superposées rapidement, soit en un endroit identique pour produire un mouvement ne nécessitant pas un déplacement de l'objet (agiter les bras, par exemple), soit avec un léger décalage s'il faut produire un déplacement (marcher). Le nombre de ces images et la taille du décalage éventuel conditionnent la qualité de l'animation: plus le nombre de dessins est élevé et plus la distance entre ceux-ci est faible, plus le mouvement sera naturel et sans heurts. Par contre, un grand nombre de dessins nécessite une capacité mémoire élevée pour leur stockage et une puissance de calcul supérieure pour les afficher à une vitesse suffisante. Il est donc indispensable de trouver un compromis en fonction de la machine utilisée et du type d'animation désiré.

Dans OrdiThéâtre, trois types d'animation sont utilisés:

- l'animation graphique proprement dite, utilisant plusieurs images pour reproduire un mouvement. Ceci est le cas lorsqu'une marionnette agite les bras ou se déplace en marchant. Trois dessins différents sont alors utilisés (voir figure 4.6.4.) et sont affichés en alternance, soit tous au même endroit (agiter les bras), soit avec un décalage de 15 pixels (15/72 pouce) entre chaque image.

- le changement de position sans image intermédiaire. Par exemple, lorsqu'une marionnette passe de la position couchée à la position debout, le dessin représentant le personnage allongé est simplement remplacé par celui représentant le personnage debout.

- le déplacement sans animation, par exemple lorsqu'une marionnette monte ou descend. Dans ce cas, le même dessin est affiché successivement à plusieurs endroits de l'écran, situés à cinq pixels (5/72 pouce) de distance les uns des autres.

L'utilisation de ces trois techniques très simples a permis d'arriver à une animation valable tout en limitant à 19 par marionnette le nombre de dessins nécessaires pour réaliser l'ensemble des mouvements prévus dans le programme.

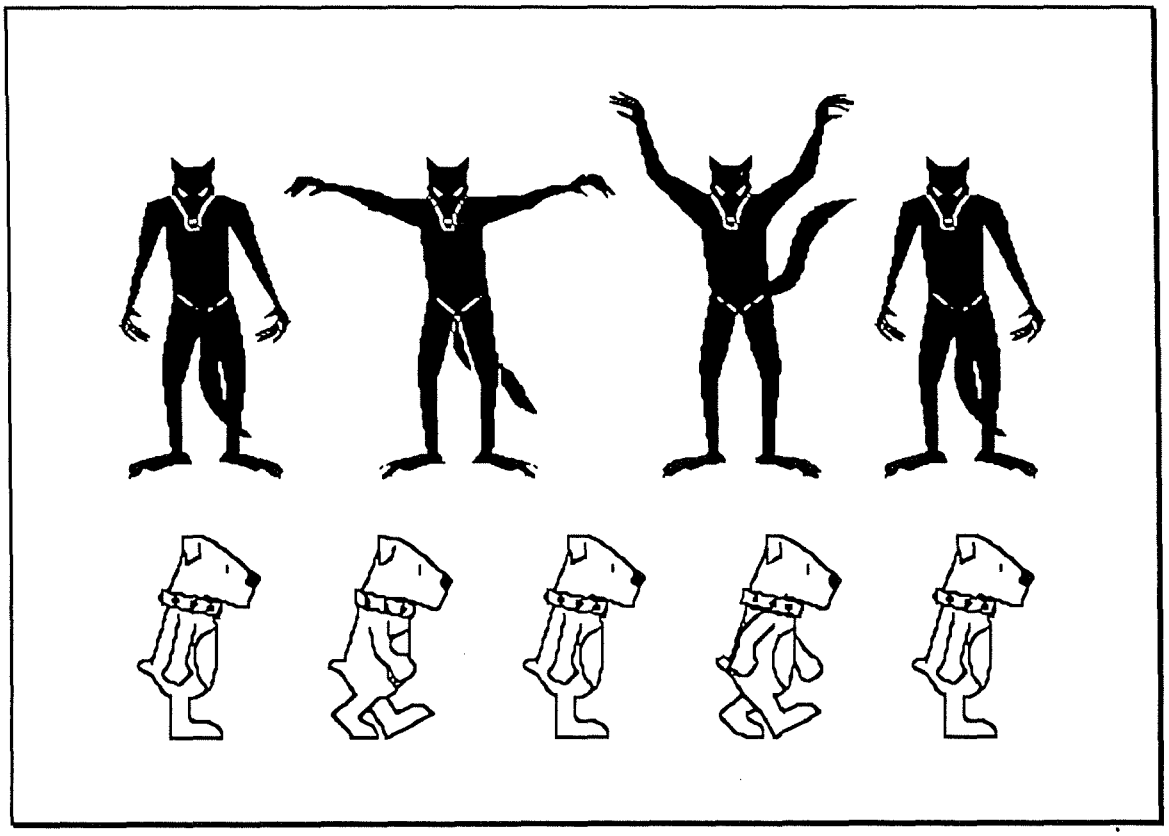


Figure 4.6.4.: l'animation graphique dans OrdiThéâtre.

6.5.2. Les techniques de composition et d'affichage des images.

Pour produire une bonne animation graphique sur un ordinateur, il est bien souvent impossible de copier les images directement vers le moniteur. En effet, déplacer un objet à l'écran suppose que l'on utilise une des deux techniques suivantes:

- soit l'objet à déplacer est d'abord effacé et ensuite recopié un peu plus loin, ce qui provoque un clignotement très gênant de l'image dû à la disparition de l'objet suivie de sa réapparition,

- soit l'objet à déplacer est d'abord recopié un peu plus loin avant d'être effacé de sa position initiale. Cette deuxième méthode provoque un dédoublement momentané de l'objet, ce qui est également fort gênant.

Pour éviter ces problèmes, nous avons opté pour une technique de recomposition en mémoire centrale de l'ensemble de l'image à afficher (décor et personnages). Celle-ci est ensuite envoyée en une seule fois vers l'écran où elle recouvre entièrement l'image précédente, exactement comme cela se passe dans une télévision. L'utilisation de graphiques sous forme de bitmaps et de fonctions de transfert de blocs en mémoire centrale particulièrement performantes rend cette méthode suffisamment rapide pour permettre une animation sans clignotements ou autres effets désagréables.

En plus des bitmaps représentant les différents dessins des personnages, la technique que nous avons développée nécessite l'utilisation de deux zones de mémoire capables de contenir un écran entier (soit 21888 octets): la première contient le décor seul et la deuxième est la zone de travail où les images sont reconstituées. L'algorithme - très simple - mis en œuvre lors de chaque étape de l'animation d'une marionnette est le suivant:

- nettoyer la zone de travail en y copiant le décor,

- copier tous les personnages nécessaires dans la zone de travail, en commençant par ceux devant se situer le plus en arrière-plan,

- copier la zone de travail dans la mémoire d'affichage.

Cette méthode permet une animation parfaite au niveau de l'affichage des différentes images, mais dont la rapidité se dégrade lorsque le nombre et la taille des objets à animer augmentent. Avec OrdiThéâtre, par exemple, la taille des personnages rendrait cet algorithme trop lent s'il fallait pouvoir animer plus de quatre marionnettes à la fois. Pour tenter de supprimer ce défaut, nous avons essayé de ne recopier dans la zone de travail que les parties de décors et les personnages affectés par le déplacement. Cependant, les nombreux tests nécessaires rendaient cette technique plus lente que celle que nous venons de présenter et que nous avons donc décidé d'utiliser.

6.6. La musique.

Pour jouer des thèmes musicaux pendant l'animation, nous avons utilisé des sons digitalisés. Cette méthode très simple permet d'obtenir des sons d'excellente qualité facilement réutilisables. Le premier travail dans ce domaine a donc consisté en la numérisation de passages musicaux provenant de disques pour enfants (des disques de Henry Dès et Christian Merveille). Ceci se fait à l'aide d'un logiciel particulier et d'un boîtier hardware se plaçant entre le Macintosh et n'importe quel matériel audio. Le logiciel permet ensuite de retravailler les musiques et de les stocker sous forme de ressources facilement réutilisables. Cependant, les sons digitalisés prennent beaucoup de place: de 5,5 kilobytes par seconde jusqu'à 22,7 kilobytes par seconde en fonction de la qualité désirée, et ce à condition de travailler en monophonie (pour du son stéréo, les capacités nécessaires sont évidemment doublées). Pour utiliser de la musique digitalisée dans OrdiThéâtre, nous avons donc dû nous contenter d'une qualité - déjà très bonne - nécessitant 7,5 kilobytes par seconde de son. La longueur des thèmes musicaux a été également limitée à une vingtaine de secondes, ce qui nous a permis de ne pas dépasser 150 kilobytes par thème.

Une fois que les sons digitalisés ont été stockés sous forme de ressources, il est très facile de les réutiliser dans un programme, et ce

grâce à des fonctions contenues dans la boîte à outils. D'autres fonctions permettent d'utiliser des sons digitalisés comme instruments et de définir des notes à jouer sur ceux-ci. Cette technique donne d'excellents résultats et consomme beaucoup moins de mémoire, mais elle nécessite l'utilisation d'éditeurs musicaux pour composer aisément les différentes mélodies. Comme les éditeurs dont nous disposions ne nous permettaient pas de récupérer les compositions réalisées avec leur concours et qu'il ne nous était pas possible de développer notre propre éditeur musical, nous avons préféré digitaliser des thèmes complets que l'on joue de manière cyclique.

Ceci termine la présentation de l'architecture logique d'OrdiThéâtre et de ses spécifications, la description de l'architecture physique et de l'environnement de programmation ainsi que l'explication des techniques d'animation graphique et musicale. Il nous reste à exposer les propositions d'améliorations de notre logiciel.

7. Propositions d'améliorations.

Le logiciel, une fois terminé, a été présenté à de nombreuses personnes et testé dans différentes institutions. Ces présentations ont provoqué de multiples réactions et des remarques concernant des possibilités d'améliorations d'OrdiThéâtre. Nous exposons ici les plus intéressantes et les plus fréquemment exprimées.

L'entrée du nom de l'enfant et des noms de scènes nécessite l'utilisation de l'entièreté du clavier. Il serait donc préférable de proposer un moyen permettant d'entrer les chaînes de caractères avec la souris ou au moyen d'un interrupteur. Ceci pourrait se faire en affichant une représentation du clavier sur l'écran, les différents caractères pouvant alors être sélectionnés avec la souris ou par un système de balayage, en fonction du mode d'entrée spécifié par les paramètres.

La possibilité de jouer à deux simultanément avait été envisagée, mais nous n'avons pas eu le temps de la développer. Les menus utilisés sont peu compatibles avec un tel mode de jeu, mais il serait possible de proposer un quatrième type d'interaction (les trois premiers modes étant

la souris, le clavier-flèches et le clavier-défilement) utilisant l'ensemble du clavier. Ce dernier serait alors divisé en deux parties, une pour chaque joueur, et à chaque touche correspondrait une action particulière. Une autre solution serait d'utiliser un deuxième ordinateur, relié au premier par un câble de communication. Chaque enfant aurait alors sa machine, qui entrerait en contact avec celle de son partenaire de jeu de manière à répercuter les actions sur les deux écrans en même temps. Cette technique est faisable tout en conservant l'interface actuelle, mais elle demande deux fois plus de matériel.

L'utilisation de la couleur serait également un plus indéniable pour le logiciel. Les modifications à apporter au logiciel sont relativement minimes, mais l'ensemble des dessins doit être colorié. Cependant, le principal obstacle à une version couleur d'OrdiThéâtre reste le prix très élevé des Macintosh avec carte et écran couleur. L'utilisation d'un ordinateur différent (Apple II GS ou Commodore Amiga par exemple) demanderait par contre la réécriture quasiment complète du logiciel; les raisons pour lesquelles nous n'avons pas utilisé une de ces machines ont été expliquées au chapitre 4.

Il nous a également été demandé s'il serait possible pour un enfant de manipuler des objets du décors, voire de créer lui-même ses décors à partir d'objets de base. Le programme actuel permettrait, sans aucune modification du code (en ajoutant simplement quelques dessins), de substituer des objets à des personnages, permettant ainsi de les déplacer et de les manipuler. Cependant, toute opération plus complexe demanderait de profonds changements du programme et rendrait plus difficile l'ajout de décors.

Un problème plus important a été soulevé lors de l'utilisation d'OrdiThéâtre: quand plusieurs marionnettes identiques ont été sélectionnées, il est très difficile lors de l'animation de les différencier. Une amélioration importante pourrait donc être réalisée en concevant un système permettant de "marquer" les marionnettes identiques se trouvant sur la scène, par exemple en leur ajoutant un numéro.

D'autres modifications et améliorations pourraient bien entendu être imaginées, et cette liste est bien entendu incomplète. Cependant, nous

pensons avoir signalé ici celles qui nous semblaient les plus pertinentes et les plus demandées.

Conclusion.

Nous avons développé ce logiciel en nous appuyant sur des méthodes de conception classiques que nous avons quelque peu adaptées pour tenter de répondre aux besoins des enfants IMC. L'évaluation menée dans différentes institutions montre le bien fondé des critères de conception d'interface que nous avons dégagés. En effet, OrdiThéâtre a pu être utilisé avec succès par la majeure partie de la population visée. Maintenant que vous connaissez OrdiThéâtre, il ne vous reste plus qu'à jouer... Bon amusement !

Conclusion.

Nous avons montré, au cours de l'introduction, combien il est important de fournir aux personnes handicapées physiques des moyens d'expression et de communication. Nous avons également vu que la technologie, et en particulier l'informatique, offrait de nombreuses possibilités dans ce domaine. Le but premier du logiciel OrdiThéâtre, réalisé au cours de ce mémoire, est de permettre à des enfants infirmes moteurs cérébraux de s'exprimer et de communiquer par le jeu de marionnettes. Il est nécessaire de proposer à ces enfants, n'ayant pas les capacités motrices pour utiliser un véritable théâtre, des substituts auxquels ils ont accès. Les utilisations espérées de ce logiciel étaient variées : utilisation uniquement ludique, ou bien pédagogique et même thérapeutique. Ces espoirs ont-ils été comblés ? Les buts visés ont-ils été atteints ?

Il nous faut dire, en premier lieu, que nous avons vécu dans le cadre de ce mémoire une expérience très intéressante et enrichissante. Nous avons appris à connaître et à travailler avec des enfants dont on entend souvent parler mais qu'en fait on ne connaît pas réellement. Les problèmes de compréhension, auxquels nous nous sommes heurtés au début de notre contact avec eux, nous ont obligé à nous adapter et nous ont montré à quel point ces enfants ont besoin de matériel spécialisé et conçu en fonction de leurs handicaps. C'est pourquoi nous nous sommes efforcés, en observant leurs difficultés d'accès à l'ordinateur et leurs capacités diverses, de créer un logiciel adapté et accessible à un maximum de ces enfants.

Nous pouvons dire que cet objectif a été atteint car tous les enfants auxquels nous avons proposé OrdiThéâtre ont pu y jouer, même si parfois son utilisation nécessitait quelques aménagements en raison des difficultés d'accès à la machine. Néanmoins tous ces enfants, âgés de 5 à 11 ans, ont compris la technique d'animation des marionnettes sur l'écran. Ils ont pu, au travers des scènes créées, s'exprimer et communiquer avec leur moniteur. Dans plusieurs cas, cette expression de leurs pensées et de leurs

sentiments a permis la détection de problèmes psychologiques qui n'avaient pas été perçus par les moyens habituels. OrdiThéâtre peut ainsi être utilisé dans un but thérapeutique. Vu la joie qu'ont procuré les marionnettes aux enfants et les demandes répétées pour "rejouer à l'ordinateur", il est évident que ceux-ci ont apprécié et ont été fortement intéressé par le logiciel. Les sourires sur les visages et les éclats de rire ont montré qu'OrdiThéatre pouvait être considéré comme un jeu offrant des moments de détente et de gaieté à des enfants qui se trouvent souvent dans des situations difficiles. Le déplacement des marionnettes et les diverses actions qu'elles peuvent réaliser permettent une utilisation pédagogique : certains apprentissages ont ainsi pu être réalisé en ce qui concerne la structuration de l'espace.

En conclusion, OrdiThéâtre a atteint les objectifs espérés au départ. Bien entendu, même si les avis ont été tous favorables, des améliorations peuvent y être apportées. Certaines améliorations concernent la modification de points précis du logiciel, mais il s'agit surtout de propositions de continuation d'OrdiThéâtre, ce qui revient à augmenter ses possibilités par des options de jeu supplémentaires. Celles-ci pourraient être réalisées dans le cadre d'un nouveau mémoire. Nous espérons également que le logiciel pourra être utilisé par une population plus étendue et qu'il pourra, par exemple, être adapté pour des enfants handicapés mentaux.

En ce qui nous concerne, nous avons été passionnés par ce travail qui a été très enrichissant aussi bien au niveau informatique qu'au niveau des contacts humains établis durant le stage.

Bibliographie.

Apple pour tous : Guide des Micro-Ordinateurs dans l'Education Spécialisée.
Apple Computer France.

Bolduc Nicole (1981) : Les jouets, bien les choisir pour amuser et développer son enfant.
Les éditions de l'homme.

Buxton W. - *Shein* F. - *Scadden* L. - *Rosen* M. - *Vanderheiden* G. (1986) : Human interface design and the handicapped user.
ACM CHI'86 Conference Proceedings, April 1986.

Chernicoff Stephen. (1987): The Macintosh revealed
Volume one : unlocking the toolbox, Hayden Books, Apple Press.
Hasbrouck Heights, New Jersey, USA, 1985.

Duff B.D. (1985): A la découverte de Macintosh.
Mc Graw-Hill.

Foley J.D. - *Wallace* V.L. - *Chan* P. (1984): The human factors of computer graphic interaction techniques.
IEEE Computer Graphics and Applications, November 1984.

Fraiture M. (FNDP) - *Machgeels* C. (ULB). (juillet 1990) : Le cahier des charges d'un logiciel pour des personnes handicapées.
Hantépsycom, Kerpape.

Hartson R. H. - *Hix* D. (March 1989) : Human computer interface development : concepts ans systems for its management.
ACM computing surveys, vol 21, N° 1.

Henniaux M. (Septembre 1981) : L'apparition de l'image expressive chez l'enfant débile par le recours à la marionnette.
Extrait de 'L'image expressive', association des psychopathologues, université de Lille III, n° spécial, p 141.

Hurtig M. et *Rondal* J-A. : Introduction à la psychologie de l'enfant.
Pierre Mardaga, tome 3.

Molich R. - *Nielsen* J. (March 1990) : Improving a human-computer dialogue.
Communications of the ACM, vol 33, N° 3.

Piaget J. et *Inhelder* B. (1986) : La psychologie de l'enfant.
Presses universitaires de France, 12ème édition.

Robaye F. (1975) : L'enfant au cerveau blessé.
Editions Dessart et Mardaga, 1975.

Ruche Jean-Claude. (1987-1988) : Méthodologie de spécification d'une interface homme-machine.
Mémoire à l'institut d'informatique.

Sand M. et *Lucas-Steinback* M-C. (1984) : Et si on jouait...?
Commission Française de la culture de l'agglomération de Bruxelles
Editeur responsable : André Ticot.

Scapin D. : Guide ergonomique de conception des interfaces homme-ordinateur.
Institut national de recherche en informatique et en automatique.

Shneiderman Ben. (1987) : Designing the user interface : strategies for effective human-computer interaction.
Addison-Wesley Publ. Comp.

THINK Technologies (1988) : THINK's Lightspeed Pascal User's Manual.
Symantec Corporation.

Vandenplas - Holper C. (1987) : Education et développement social de l'enfant.
Presses universitaires de France, 2ème édition.

van Lamsweerde A. (1988) : Méthodologie de développement de logiciels.
Notes de cours (F.U.N.D.P, Institut d'Informatique).

Verburg G. - *Field* D. - *St. Pierre* F. - *Naumann* S. (1987): Towards universality of access: interfacing physically disabled students to the ICON educational microcomputer.
ACM CHI + GI 1987 Conference Proceedings, April 1987.

Ware C. - *Harutune* H. M. (1987): An Evaluation of an Eye Tracker as a Device for Computer Input.
ACM CHI + GI 1987 Conference Proceedings, April 1987.

Facultés Universitaires Notre-Dame de la Paix
NAMUR

Institut d'Informatique

Logiciel de simulation d'un jeu de marionnettes pour enfants infirmes moteurs cérébraux

Annexes

Laurence Brousmiche
Xavier Gillet

Promoteurs: M. Noirhomme-Fraiture
M. Mercier

Mémoire présenté dans le but de l'obtention du titre de
Licencié et Maître en Informatique
1989 - 1990

Table des matières.

ANNEXE 1 :

Manuel d'utilisation d'OrdiThéâtre.

TABLE DES MATIERES.

INTRODUCTION.

1. Cadre de conception du logiciel.

Le logiciel ORDITHEATRE a été réalisé dans le cadre de notre mémoire de fin d'études, sous la direction de Madame Noirhomme (FNDP, institut d'informatique) et de Monsieur Mercier (FNDP, département de psychologie). Ce sujet avait été proposé par Mr Mercier en collaboration avec le service IMC du centre hospitalier universitaire de Kremlin-Bicêtre (Paris). C'est pourquoi OrdiThéâtre a été conçu au départ pour des enfants infirmes moteurs cérébraux âgés de 6 à 12 ans, mais nous espérons qu'il sera utilisé par une population beaucoup plus étendue.

2. L'utilisation de l'informatique par les IMC: pourquoi et comment ?

Afin de dégager l'apport possible de l'informatique aux enfants IMC et les adaptations matérielles et logicielles nécessaires à son utilisation, il nous semble utile de décrire brièvement ce que l'on entend par IMC.

"Nous appellerons IMC celui qui a été atteint avant, pendant ou peu de temps après sa naissance, d'une anomalie non évolutive et non curable des tissus cérébraux, se manifestant entre autres par des troubles moteurs. L'Infirmité Motrice d'origine Cérébrale est donc un état pathologique et non une maladie" [1] . On distingue plusieurs types de troubles moteurs dont les deux plus importants semblent être la spasticité et l'athétose. Dans la spasticité, les muscles atteints sont raides et hyperexcitables. Lorsqu'un mouvement est ébauché, les muscles qui

---

[1] "L'enfant au cerveau blessé", Francine Robaye, p13.

devraient se relâcher n'y parviennent pas, ce qui fait que le mouvement s'arrête à mi-chemin ou est paralysé. Un athétosique est agité de mouvements spasmodiques involontaires, incontrôlés et incoordonnés, qui rendent l'action maladroite et difficile. Ces troubles peuvent être combinés avec d'autres. Ils sont de gravité variable et se situent à différents niveaux (membres inférieurs et supérieurs, tête, face, ...) En conséquence, nous nous trouvons devant des enfants dont les difficultés sont multiples et variées. Cela peut aller de problèmes dans l'utilisation des bras ou des jambes jusqu'à la paralysie quasi-totale, en passant par des troubles de la parole dus à des problèmes moteur au niveau de la face (impossibilité de former les sons du langage). A ceci s'ajoutent des troubles de la vue et de l'audition à des degrés divers. Le fonctionnement intellectuel global de l'IMC peut varier entre la débilité mentale profonde jusqu'à l'intelligence supérieure et brillante.

Vu la diversité des troubles, le matériel nécessaire à l'utilisation de l'ordinateur par un maximum de ces enfants doit être spécialisé et étendu. L'accès à la machine se fait par le biais de licorne, mini-clavier, casque ultrasonique (remplaçant la souris), clavier étendu, interrupteurs de différents types,... Le logiciel doit aussi tenir compte de ces modes d'entrée en présentant une interface élaborée. Celle-ci doit permettre l'emploi d'un seul bouton (système à balayage), aussi bien que la manipulation directe de la souris ou du clavier.

Si l'on tient compte de ces adaptations, l'ordinateur devient un outil à la portée d'un grand nombre de ces enfants. Mais il nécessite un temps d'adaptation et d'apprentissage et donc des efforts parfois considérables. Il faut donc bien en évaluer les avantages et les inconvénients. Il ne faut pas à tout prix vouloir utiliser l'ordinateur pour réaliser une activité qui, effectuée avec un crayon et une feuille de papier, apporterait autant à l'enfant. De plus, le coût d'une installation informatique n'est pas négligeable. Cependant, lorsque cela en vaut la peine, une aide informatique bien pensée permet une ouverture vers des possibilités nouvelles. L'ordinateur peut ainsi devenir un outil-prothèse et permettre la réalisation d'actions autrement impossibles. Par exemple, un logiciel de synthèse vocale facilite l'expression et donne la parole à des enfants qui en sont souvent privés. En plus de cet aspect prothèse,

l'informatique peut fournir un apport pédagogique, culturel ou ludique. Il existe, en effet, de nombreux programmes d'aide à l'apprentissage de la lecture, de l'écriture et du calcul; des programmes de jeux de toutes sortes, malheureusement souvent inadaptés aux IMC; ainsi que des programmes à tendance culturelle (atlas géographique par exemple). Le logiciel de marionnettes OrdiThéâtre se situe dans un cadre à la fois pédagogique, ludique et même thérapeutique.

3. Pourquoi un logiciel de simulation d'un jeu de marionnettes ?

Dans le développement de l'enfant, le jeu a une importance primordiale. Il lui apporte du plaisir, c'est un moyen d'apprentissage et d'expression. Le jeu permet à l'enfant d'exprimer ce qu'il vit, de reproduire ce qui pour lui est gratifiant ou ce qui est conflictuel et difficile. Il acquiert une maîtrise des événements en les rejouant. Lorsqu'il a été mis en échec dans la réalité, il lui arrive de rejouer la situation, en l'inversant, et cette fois en triomphant des difficultés. Il peut, par le jeu, extérioriser ses émotions : ses joies, ses peines, ses angoisses, ses frustrations, ses colères, ses désirs,... et ainsi apprendre à les dominer. Cette maîtrise est renforcée par le fait qu'il est actif, qu'il ne subit plus la réalité. Par exemple, une fillette à qui le kinésithérapeute avait placé des attelles en présence de stagiaires et qui avait très mal vécu la situation, a rejoué la scène en devenant le kinésithérapeute : elle plaçait des attelles à sa poupée tout en expliquant l'activité à ses stagiaires. En rejouant une situation où elle s'était sentie passive, totalement dépendante et manipulée, elle devient active et domine la situation en étant le kinésithérapeute. Même si elle avait repris le rôle de l'enfant, elle aurait aussi retrouvé une maîtrise puisque c'était elle maintenant le metteur en scène. Le théâtre de marionnettes présente un intérêt particulier dans l'optique de maîtrise, d'extériorisation et d'expression des sentiments de l'enfant. Tout en suscitant l'imagination et la créativité, ce jeu lui offre des marionnettes dans lesquelles il peut se projeter, ainsi que son entourage et son environnement. Par cette projection, qui est grandement facilitée par le pouvoir d'attrait des marionnettes sur l'enfant, il peut placer une distance entre lui et la réalité, les événements, les émotions, puisque ce n'est plus lui-même mais la

marionnette qui subit la situation. Par la mise en scène, il exprime certains sentiments de manière plus libre, plus naturelle, plus concrète que s'il devait les expliciter verbalement, le jeu étant le moyen naturel d'expression de l'enfant. Un observateur attentif, psychologue ou autre, peut alors "l'écouter" et ainsi recevoir ce que l'enfant a à dire. Un autre aspect intéressant des marionnettes est qu'elles permettent à l'enfant de se dépasser. En effet, il peut leur faire effectuer des actions qui lui sont, soit impossibles -par exemple, une marionnette qu'il fait tournoyer dans les airs le transformera en un fabuleux acrobate, soit interdites - plutôt que de frapper un autre enfant, il déchargera son agressivité en faisant se disputer les marionnettes.

OrdiThéâtre a pour but d'offrir ces possibilités de communication aux enfants IMC qui n'ont pas les capacités motrices pour utiliser un théâtre de marionnettes. C'est un moyen de donner à un enfant handicapé la possibilité de s'extérioriser, d'exprimer ce qu'il ressent et d'énoncer ses problèmes éventuels, et il peut donc ainsi servir de base à une thérapie. Il faut cependant se rendre compte qu'une adaptation sur ordinateur n'est pas le reflet exact d'un véritable théâtre. Certaines restrictions apparaissent : la mobilité des marionnettes est diminuée, le contact avec elles est moins direct et l'utilisation en est plus complexe. Par contre, des possibilités supplémentaires sont apportées : l'enfant peut enregistrer le déroulement de la scène et donc la visualiser par après, le moniteur dispose de la trace des manipulations de l'enfant afin de les étudier par la suite.

Enfin, il est intéressant de comparer le pouvoir d'attrait d'une "marionnette informatique" à celui d'une véritable marionnette. En bref, la simulation d'un théâtre sur ordinateur suscite-t-elle les mêmes émotions et réactions que son modèle ?

MANUEL D'UTILISATION.

1. Les paramètres.

OrdiThéâtre est une simulation, sur ordinateur, d'un jeu de marionnettes. Il permet à l'enfant de choisir un décor, une musique et des personnages, et ensuite, de créer une scène en animant les personnages dans le décor. Il offre aussi la possibilité d'enregistrer et de visualiser par après les scènes créées. Ayant été conçu pour des enfants IMC qui n'ont pas les capacités motrices nécessaires à l'utilisation d'un véritable théâtre, il comprend un certain nombre de paramètres modifiables en fonction de l'enfant.

Ces paramètres, étant fonction de l'enfant, seront donc enregistrés selon son nom (ou tout autre nom significatif). Cinq paramètres sont disponibles :

1.1. L'accès ou mode d'utilisation.

Afin d'être accessible à un maximum d'enfants, l'activation des commandes du logiciel se fait par le positionnement d'un curseur dans une ligne de commandes située dans le bas de l'écran. Cette ligne regroupe des commandes illustrées par un dessin simple et explicite. Le déplacement du curseur et la validation seront fonction du mode d'utilisation choisi, à savoir:

* la souris : déplacement du curseur à l'aide de la souris et validation en cliquant. Ce mode permet également l'utilisation d'un joystick, d'une trackball, d'un casque ultrasonique ou de tout autre matériel s'adaptant à la sortie souris.

* le clavier-flèches : déplacement du curseur dans la ligne de commandes à l'aide des flèches <-, -> ou des lettres K et L, et validation au moyen de <RETURN>. Le clavier normal peut être remplacé par un mini-clavier, un clavier étendu ou un jeu d'interrupteurs raccordés à la Magic Box.

* le clavier-défilement : déplacement automatique du curseur dans la ligne de commandes et validation au moyen de <RETURN>. La validation peut également s'effectuer par le biais d'un interrupteur si l'on utilise la Magic Box.

1.2. La vitesse de défilement.

Lorsque le mode d'utilisation sélectionné est le 'clavier-défilement', il est possible de régler la vitesse à laquelle le curseur se déplace dans la ligne de commandes.

1.3. La constitution des lignes de commandes.

Ce paramètre permet au moniteur d'organiser, comme il le désire, les lignes de commandes de la partie animation des marionnettes. En effet, à ce stade du jeu, l'enfant dispose d'un maximum de quatre lignes de commandes qu'il fait apparaître successivement dans le bas de l'écran. Le moniteur peut déterminer le nombre de lignes qu'il propose à l'enfant, ainsi que le nombre et la place des commandes sur chaque ligne. Il peut ainsi ne proposer qu'un nombre limité de commandes lors de la première utilisation (par exemple, les flèches de déplacement uniquement) et ensuite, en ajouter d'autres au fur et à mesure que l'enfant se familiarise avec OrdiThéâtre (les commandes 'assis', 'debout', 'couché' par exemple). Si l'enfant déplace les marionnettes au moyen de la souris, le moniteur peut supprimer les flèches de déplacement des lignes de commandes.

1.4. L'enregistrement automatique.

Pour pouvoir visualiser par la suite la scène qu'il crée, l'enfant doit enregistrer la scène au moment même de sa création. Pour ce faire, il existe une commande d'enregistrement illustrée par une caméra. Mais il se peut que le moniteur n'ait pas prévu de proposer cette commande à

l'enfant , par exemple lors de la première utilisation. Il est fort probable aussi, même si l'enfant dispose de cette caméra, qu'il oublie de la sélectionner. Il en résulte alors l'impossibilité de revoir la scène puisqu'elle n'a pas été enregistrée. Pour éviter cela, le moniteur a la possibilité de brancher l'enregistrement automatique. Celui-ci entraîne la mise en route de la 'caméra' dès l'entrée dans la partie animation. Si l'enfant veut alors couper cet enregistrement, il lui suffit de débrancher la 'caméra' pourvu que celle-ci lui soit proposée sur une des lignes de commandes auxquelles il a accès.

1.5. La trace.

Le dernier paramètre concerne la trace c'est-à-dire l'enregistrement de toutes les manipulations faites durant l'utilisation du logiciel (aussi bien celles de l'enfant que celles du moniteur). Cet enregistrement peut ne pas avoir lieu; ou peut avoir lieu uniquement durant la phase de sélection du décor, de la musique, des marionnettes ou de la visualisation de la scène; ou durant la phase d'animation; ou durant ces deux dernières phases.

Ces paramètres peuvent, à tout moment, être consultés, modifiés et enregistrés en fonction du nom de l'enfant. La marche à suivre pour effectuer ces opérations est exposée dans le point 4. MENUS. Pour mieux comprendre ces paramètres, découvrons d'abord le logiciel Ordithéâtre.

2. Lancement d'OrdiThéâtre.

Avant de commencer, il est préférable faire une copie des différentes disquettes fournies avec OrdiThéâtre, surtout si vous ne disposez pas d'un disque dur. Rangez ensuite soigneusement les disquettes originales et travaillez toujours avec les copies.

2.1. Systèmes sans disque dur disposant de deux lecteurs de disquettes.

Allumez l'ordinateur au moyen de l'interrupteur situé sur la gauche de la face arrière de la machine. Après un 'bip' sonore, une disquette barrée et un point d'interrogation clignotent au centre de l'écran, indiquant que votre Macintosh attend une disquette système.

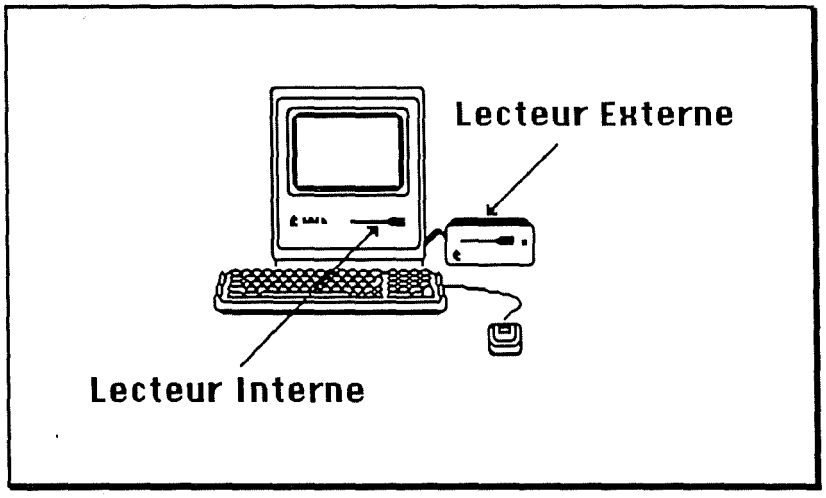


Schéma 1: Macintosh Plus équipé de deux lecteurs de disquettes.

Prenez la disquette marquée "OrdiThéâtre: Système + Personnages" et introduisez-la **dans le lecteur interne** (voir schéma 1), **l'étiquette vers le haut** et **la partie métallique vers l'ordinateur** (sens indiqué par la flèche gravée sur le bord gauche de la disquette - voir schéma 2). Un ordinateur 'souriant' apparaît au centre de l'écran: le système se charge.

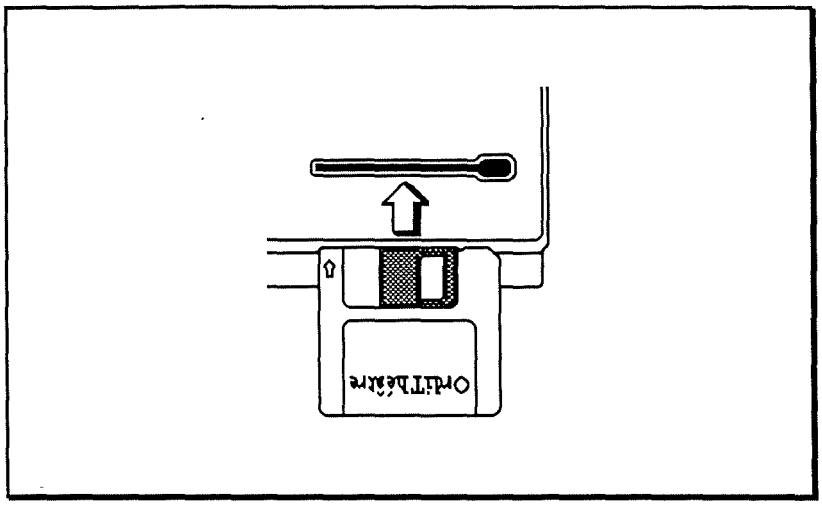


Schéma 2: introduction d'une disquette dans le lecteur.

Prenez la disquette marquée 'OrdiThéâtre: Programme + Décors + Musiques' et introduisez-la **dans le lecteur externe** (voir schéma 1), **dans le même sens que la précédente**. Après un bref instant, deux icônes représentant des disquettes doivent être présentes à l'extrême droite de l'écran: celle du dessus porte le nom 'Disque Système', celle du dessous porte le nom 'Disque OrdiThéâtre'. Déplacez le pointeur de la souris sur l'icône de la disquette 'Disque OrdiThéâtre' puis cliquez deux fois rapidement, sans bouger la souris: une fenêtre apparaît, contenant des icônes représentant le programme et les divers fichiers qu'il manipule (voir schéma 3).

Déplacez le pointeur de la souris à l'intérieur de la fenêtre et cliquez deux fois rapidement sur une des icônes qu'elle contient, mais pas sur un fichier trace (fichiers de type MacWrite dont le nom se termine par ".trace.*numéro*"). Le programme va alors démarrer et afficher pendant quelques secondes une fenêtre de présentation. Si tout s'est bien déroulé, vous pouvez maintenant passer au point 3 de ce manuel: OrdiThéâtre!

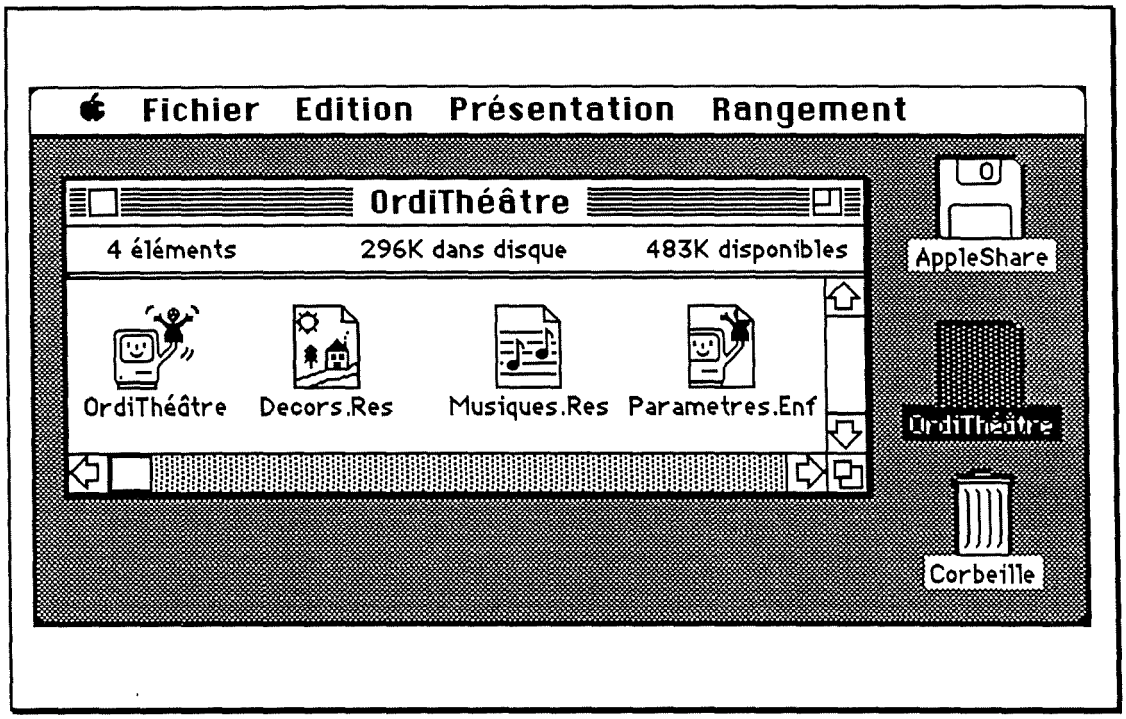


Schéma 3: le Finder et la fenêtre "OrdiThéâtre".

2.2. Systèmes équipés d'un disque dur.

Si vous disposez d'un disque dur, il vous est possible d'utiliser la version complète d'OrdiThéâtre, comprenant six thèmes musicaux au lieu de trois. Une petite installation est nécessaire avant la première utilisation du logiciel.

2.2.1. L'installation sur disque dur.

Une fois votre Macintosh allumé, "ouvrez" le disque dur en cliquant deux fois rapidement sur son icône. Sélectionnez l'option *Nouveau dossier (New folder)* dans le menu *Fichier (File):* un nouveau dossier est créé sur le disque; il porte le normalement le nom "dossier vide" (ou "empty folder"). Rennomez-le "OrdiThéâtre".

Introduisez la disquette nommée "OrdiThéâtre : Système + Personnages" dans le lecteur de disquettes et cliquez deux fois

rapidement sur l'icône "Disque Système". Copiez ensuite le fichier "Personnages.Res" sur votre disque dur, dans le dossier OrdiThéâtre, puis éjectez la disquette en faisant glisser son icône dans la poubelle.

Introduisez la disquette "OrdiThéâtre : Programme + Décors + Musiques" dans le lecteur de disquettes et cliquez deux fois rapidement sur l'icône "Disque OrdiThéâtre". Copiez les fichiers "OrdiThéâtre" et "Décor.Res" (PAS le fichier "Musiques.Res"!) sur votre disque dur, dans le dossier OrdiThéâtre, puis éjectez la disquette.

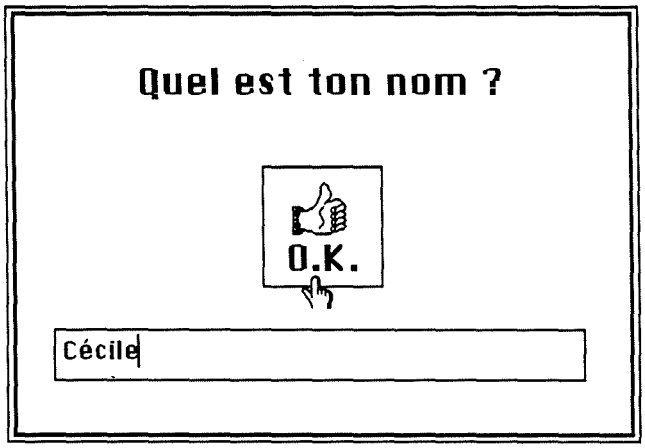
Introduisez la disquette "OrdiThéâtre : Musiques - disque dur" dans le lecteur de disquettes et cliquez deux fois rapidement sur l'icône "Disque Musiques". Copiez le fichier Musiques.Res sur votre disque dur, dans le dossier OrdiThéâtre, puis éjectez la disquette. L'installation est maintenant terminée et vous pouvez ranger les disquettes en lieu sûr: vous n'en aurez plus besoin que pour installer le programme sur un autre disque dur.

2.2.2. Démarrage du programme à partir du disque dur.

Ouvrez le dossier "OrdiThéâtre": une fenêtre apparaît, contenant des icônes représentant le programme et les divers fichiers qu'il manipule (voir schéma 3 ci-dessus). Déplacez le pointeur de la souris à l'intérieur de la fenêtre et cliquez deux fois rapidement sur une des icônes qu'elle contient, mais pas sur un fichier trace (fichiers de type MacWrite dont le nom se termine par ".trace.*numéro*"). Le programme va alors démarrer et afficher pendant quelques secondes une fenêtre de présentation.

3. OrdiThéâtre.

ECRAN 1.



Pour répondre à la question, tapez le nom de l'enfant au clavier, suivi de <RETURN> ou bien d'un clic-souris dans la case OK.

Les implications du nom introduit à ce moment sont les suivantes :

* sur les paramètres : OrdiThéâtre va chercher les paramètres correspondants au nom introduit. Par exemple, si le mode d'utilisation défini pour ce nom est le clavier-flèches, un curseur apparaîtra dans la ligne de commandes du bas de l'écran, dès l'écran suivant. L'enfant pourra déplacer le curseur à l'aide des flèches et valider sa commande par <return>. Si aucun paramètre n'a été enregistré sous ce nom ou si le logiciel ne reconnaît pas ce nom (à cause d'une erreur de frappe par exemple), il prend les paramètres définis par défaut. Ces paramètres par défaut sont :

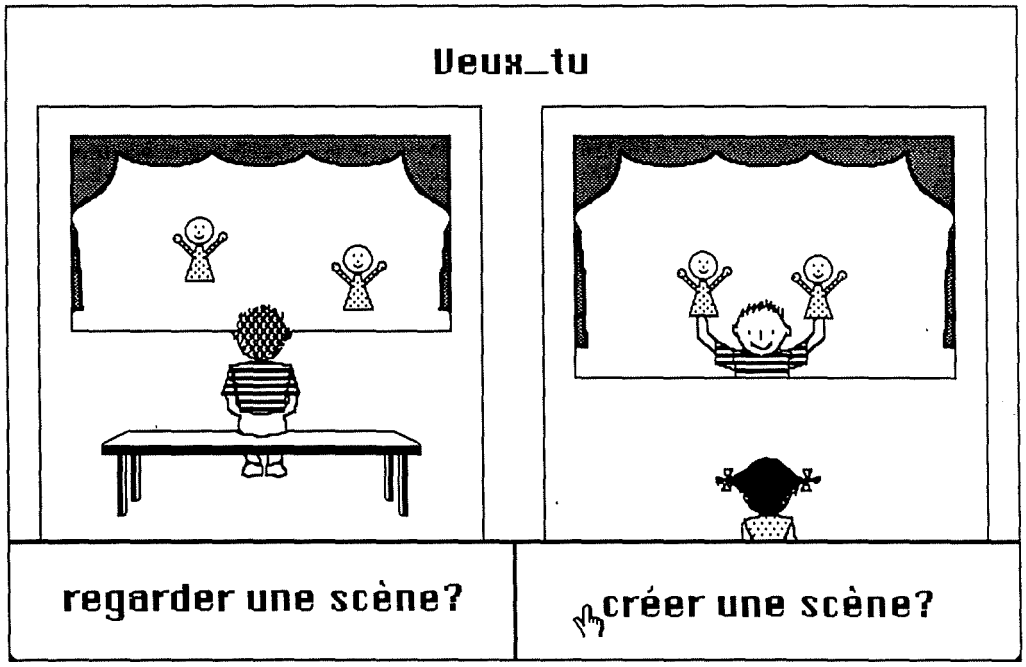
- mode d'utilisation : la souris;
- (vitesse de défilement : 1.5 sec);

- lignes de commandes : toutes les commandes sont disponibles sur 3 lignes;
- enregistrement automatique : actionné;
- trace : pas de trace.

* sur les scènes enregistrées : lorsque l'enfant voudra regarder les scènes créées précédemment, seules les scènes enregistrées sous son nom lui seront proposées. Il ne peut donc pas regarder les scènes d'autres enfants. Ceci permet d'éviter qu'il ne modifie ou même qu'il ne détruise des scènes ne lui appartenant pas.

* sur la scène qu'il crée : si l'enfant enregistre une scène, elle le sera sous le nom introduit ici.

ECRAN 2.

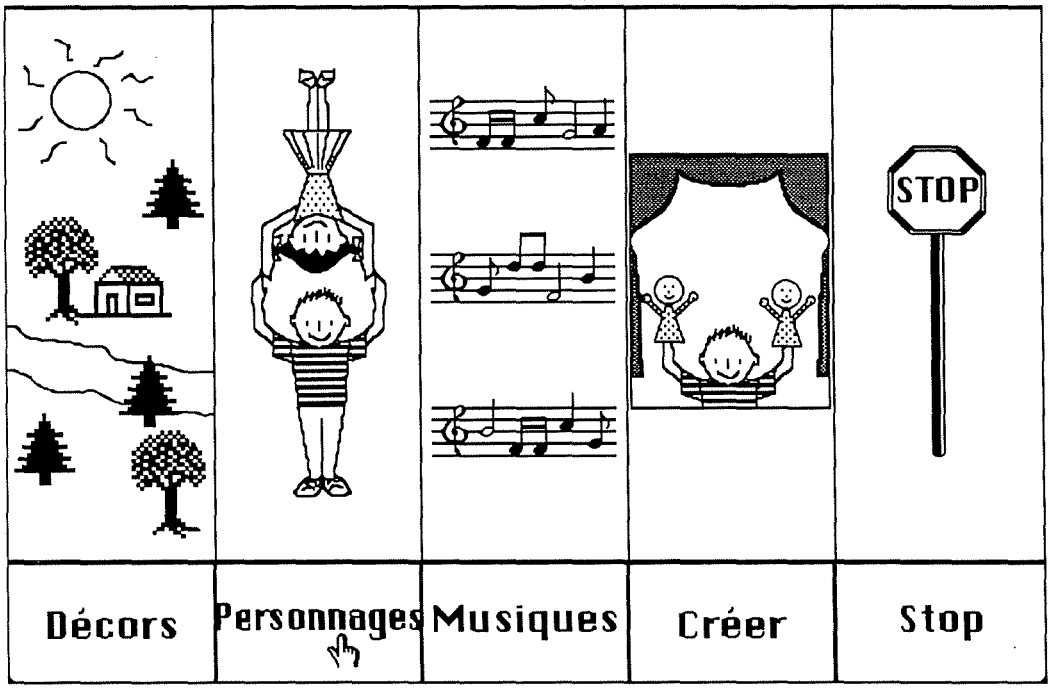


A partir de ce moment, l'enfant peut utiliser seul le logiciel. Sur cet écran lui sont présentées les deux grandes options d'OrdiThéâtre : REGARDER UNE SCENE ou CREER UNE SCENE. Pour marquer son choix, l'enfant place le curseur sur une des deux cases du bas de l'écran et valide selon son mode d'utilisation (<return> ou clic-souris).

Créer une scène ---> ECRAN 3.
Regarder une scène ---> ECRAN 9.

Option 'CREER UNE SCENE'.

ECRAN 3.



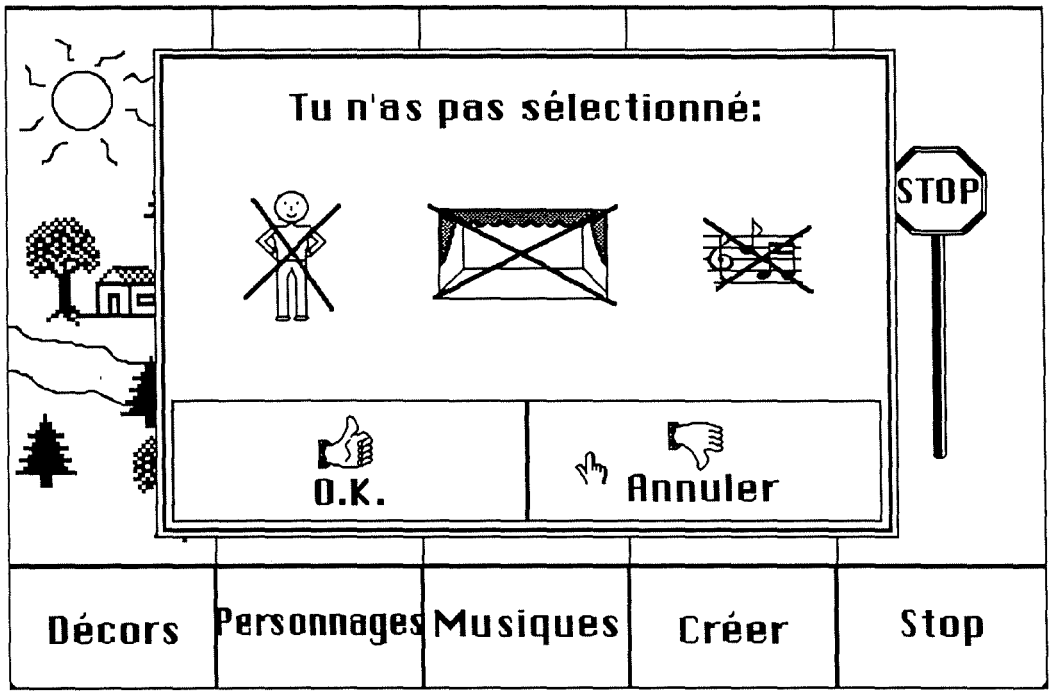
Après la validation de la commande 'créer une scène' de l'écran 2, on passe à la sélection d'un décor, des personnages et/ou d'une musique. L'enfant choisi son décor, ses marionnettes et sa musique dans l'ordre qu'il préfère. Il peut aussi ne pas sélectionner de décor ou de musique. La sélection de l'option voulue se fait toujours en plaçant le curseur dans une des cinq cases du bas de l'écran et en validant par <return> ou un clic-souris.

Case 1 ---> sélection d'un décor, passage à l'ECRAN 4.
Case 2 ---> sélection des personnages, passage à l'ECRAN 5.
Case 3 ---> sélection d'une musique, passage à l'ECRAN 7.

Case 4 ---> animation, passage à l'ECRAN 8 ou apparition de la FENETRE 1.

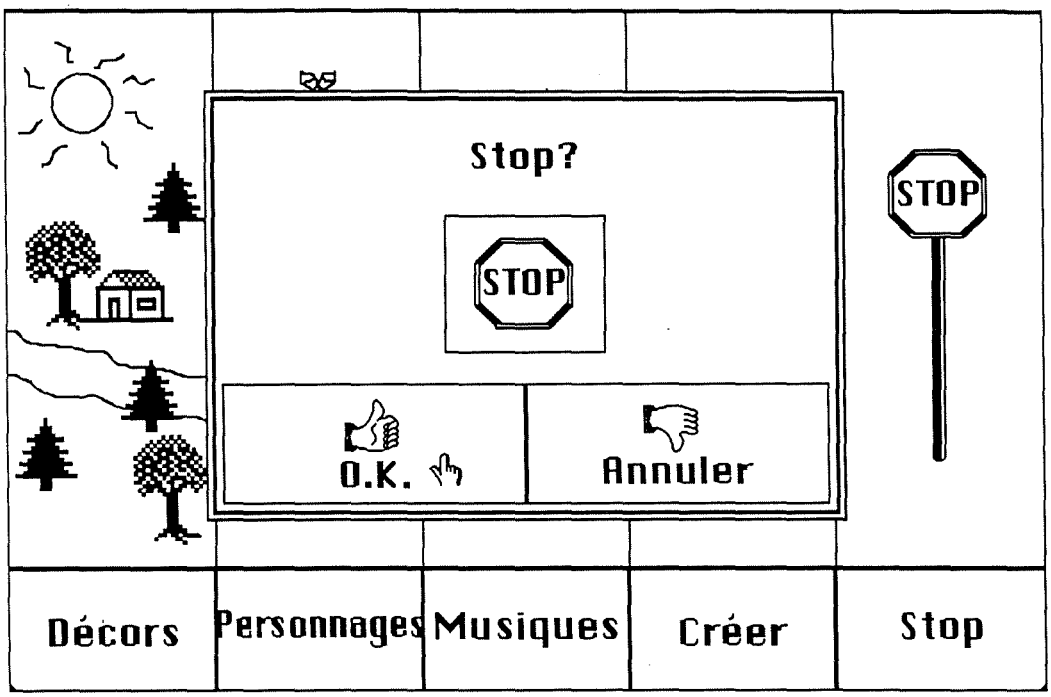
Case 5 ---> stop,retour à l'ECRAN 2 avec apparition préalable de la FENETRE 2.

FENETRE 1 :



Cette fenêtre indique à l'enfant qu'il n'a pas sélectionné de décor et/ou de personnage et/ou de musique. Si cela était volontaire et qu'il désire jouer sans décor par exemple, il valide la commande OK. Sinon, il a la possibilité de rester à l'écran 3 en validant la commande ANNULER.

FENETRE 2 :

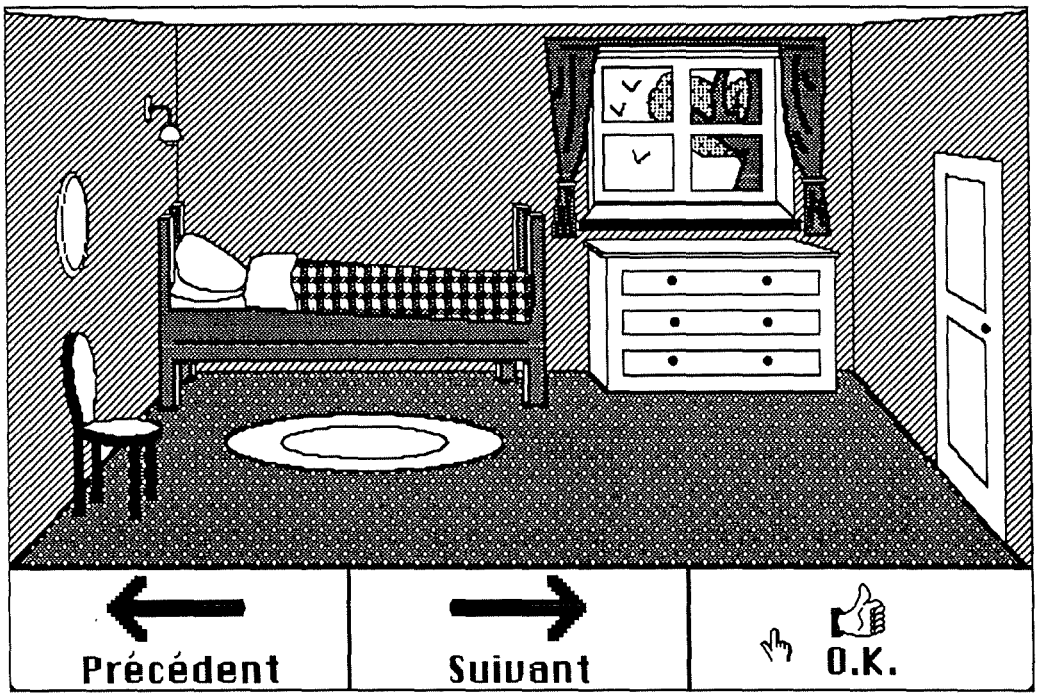


Cette fenêtre est une demande de confirmation de la commande STOP. En effet, cette commande implique l'annulation de tout ce qui a déjà été sélectionné. Si l'enfant a déjà choisi son décor et qu'il retourne à l'écran 2, il devra de nouveau le sélectionner lorsqu'il reviendra à l'écran 3 par la suite. C'est pourquoi une confirmation est demandée.

La validation de la commande OK entraîne le retour à l'écran 2 avec perte des sélections déjà effectuées.

La validation de la commande ANNULER entraîne le retour à l'écran 3 sans perte des sélections.

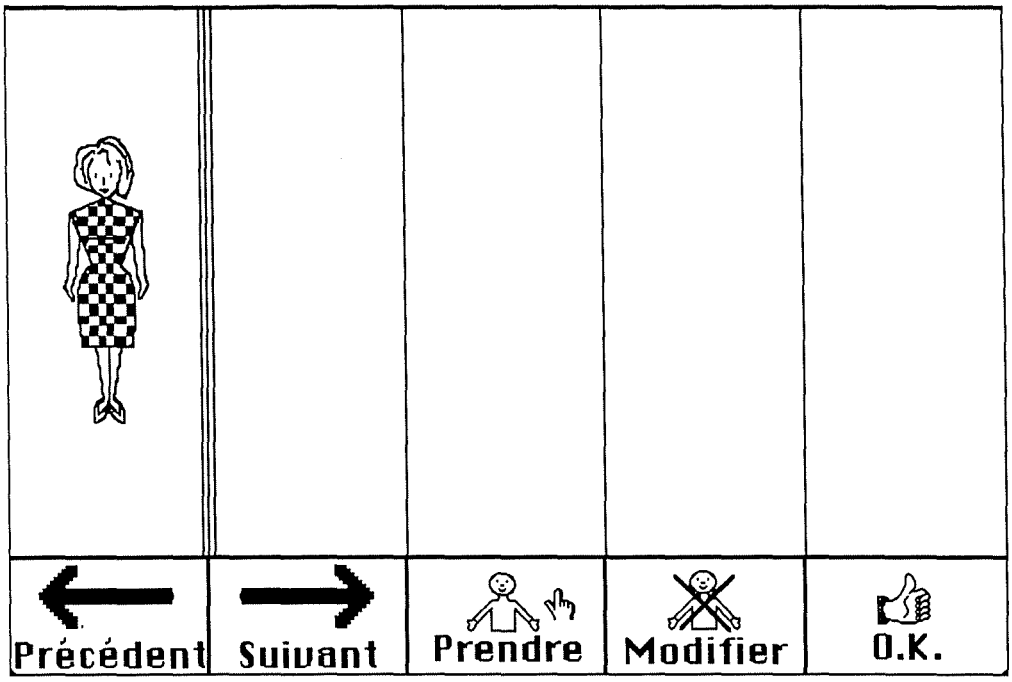
ECRAN 4.



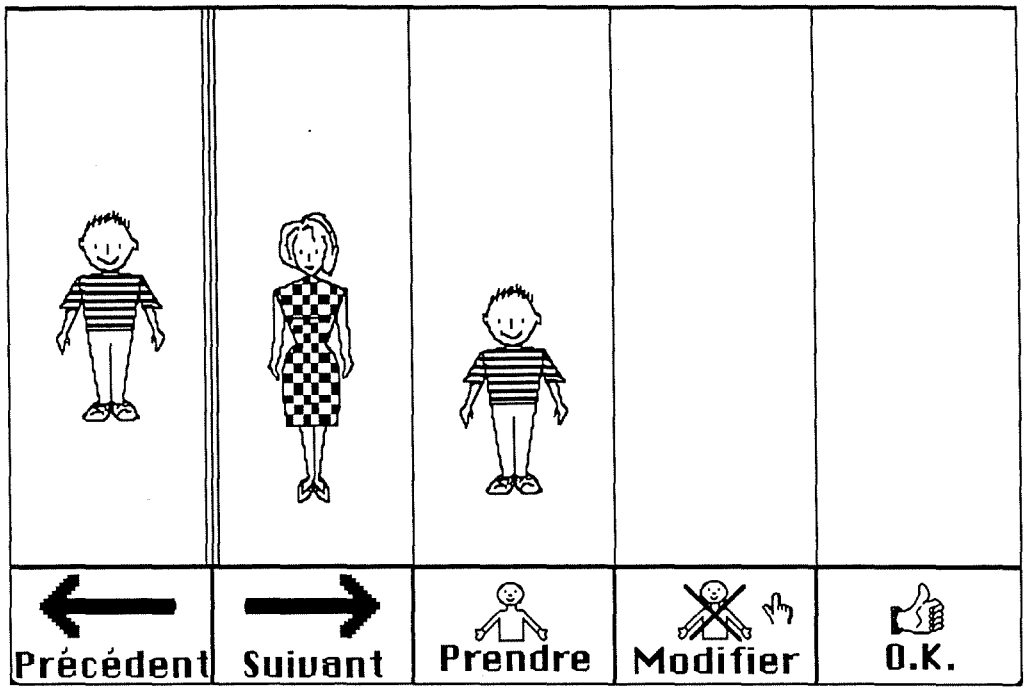
Cet écran concerne la sélection d'un décor. La visualisation des différents décors s'effectue au moyen des commandes PRECEDENT et SUIVANT. Le déroulement est circulaire : lorsque l'on est sur le dernier décor et que l'on fait 'suivant', on revient au premier; de même lorsque l'on est face au premier décor et que l'on passe au décor précédent, on arrive au dernier. Quand l'enfant est face au décor qu'il veut choisir, il valide ce choix par la commande OK.

Case 1 ---> passage au décor précédent.
Case 2 ---> passage au décor suivant.
Case 3 ---> sélection du décor et retour à l'écran 3.

ECRAN 5.



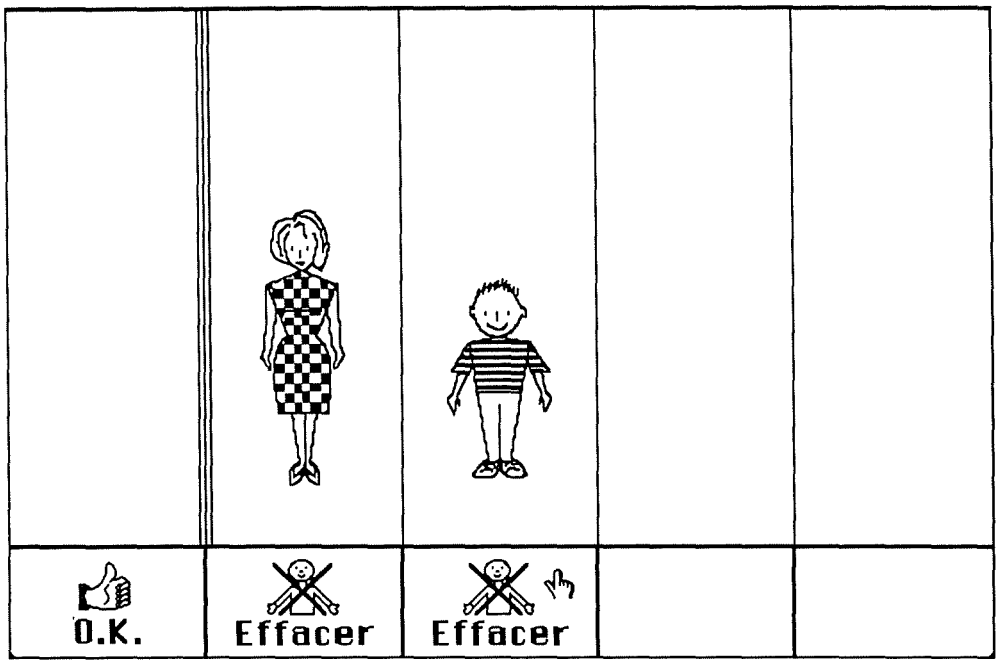
Cet écran permet de choisir un maximum de 4 marionnettes qui seront animées lors de la création de la scène. Dans la case supérieure gauche sont présentées, une à une, les différentes marionnettes disponibles. Dans les quatre cases de droite, les marionnettes s'affichent au fur et à mesure de leur sélection. Par exemple, dans l'écran ci-après, la maman et le garçon ont déjà été choisis.



Il n'existe aucune restriction quant aux marionnettes choisies : l'enfant peut prendre 2, 3, et même 4 fois la même s'il le désire. Le défilement dans la case de gauche s'effectue au moyen des commandes PRECEDENT et SUIVANT. Le déroulement des personnages y est circulaire. Pour sélectionner une marionnette, il faut valider la commande PRENDRE. La marionnette s'affiche alors dans une des quatre cases de droite. Il est possible de modifier son choix en cours de route et donc de "désélectionner" une marionnette. Ceci s'effectue au moyen de la commande MODIFIER. Celle-ci entraîne le passage à l'écran de modification (écran 6) et permet la suppression d'une ou de plusieurs marionnette(s) déjà sélectionnée(s). Lorsque le choix des personnages est définitif, la commande OK valide ce choix.

Case 1 ---> marionnette précédente.
Case 2 ---> marionnette suivante.
Case 3 ---> sélection de la marionnette.
Case 4 ---> modification, passage à l'écran 6.
Case 5 ---> validation du choix définitif.

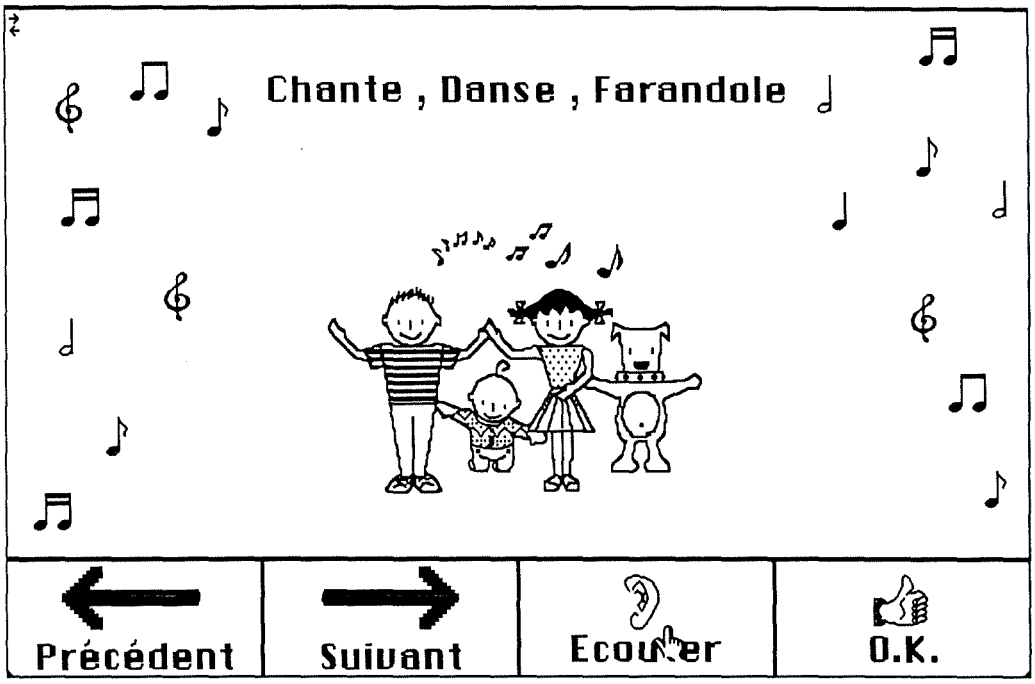
ECRAN 6.



Cet écran concerne la suppression d'une ou de plusieurs marionnette(s) sélectionnée(s). Celle(s)-ci apparaît(ssent) dans les quatre cases de droite. Pour effacer une marionnette, il faut valider la commande EFFACER située dessous. Lorsque le choix des marionnettes est correct, la commande OK permet de revenir à l'écran 5 et d'en choisir de nouvelles.

Case 1 ---> retour à l'écran 5;
Cases 2,3,4 et 5 ---> suppression de la marionnette située au-dessus de la commande.

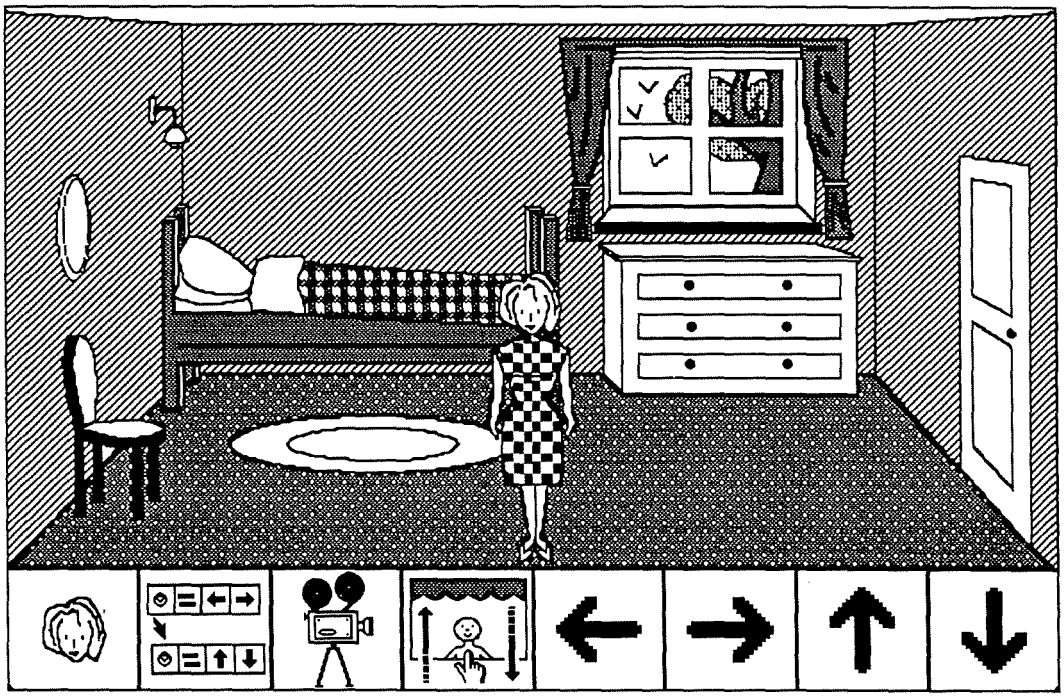
ECRAN 7.



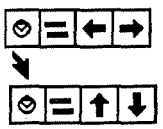
Cet écran concerne la sélection d'une musique. Les commandes PRECEDENT et SUIVANT permettent de passer en revue les différentes musiques disponibles. Le déroulement est circulaire (idem que pour les décors). La commande ECOUTER permet l'arrêt ou la mise en route de la musique. Quand l'enfant entend la musique qu'il veut choisir, il valide ce choix par la commande OK.

Case 1 ---> passage à la musique précédente.
Case 2 ---> passage à la musique suivante.
Case 3 ---> arrêt ou mise en route de la musique.
Case 4 ---> sélection de la musique et retour à l'écran 3.

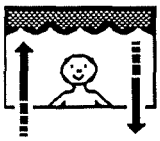
ECRAN 8 : Partie animation.



On se trouve ici dans la partie d'animation. C'est ici que l'enfant crée une scène en animant ses marionnettes dans le décor et au son de la musique. Pour créer cette scène, l'enfant valide les commandes qui lui sont proposées dans le bas de l'écran. Au départ, l'écran comprend uniquement le décor et la première ligne de commandes. Les marionnettes sont 'en coulisse' et c'est à l'enfant de les faire entrer en scène. Le principe est que l'enfant commande toujours un personnage à la fois, celui dont la tête est présentée dans la première case (en bas, à gauche de l'écran). C'est ce personnage qui effectuera la commande choisie. Pour changer, il faut cliquer sur cette case et la tête d'une autre marionnette apparaît. Les commandes sont présentées sur plusieurs lignes (au maximum quatre) que l'enfant peut faire apparaître successivement. Les commandes <caméra> et <musique> s'affichent en inverse lorsqu'elles sont activées. Les différentes commandes disponibles sont :



: passage à la ligne de commandes suivante.



: entrée ou sortie de scène.

: la marionnette se déplace d'une distance donnée vers la gauche.

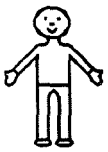
: la marionnette se déplace d'une distance donnée vers la droite.

: la marionnette se déplace d'une distance donnée vers le haut.

: la marionnette se déplace d'une distance donnée vers le bas.

: la marionnette s'assied.

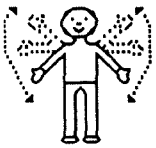
: la marionnette se couche.



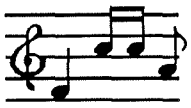
: la marionnette se met debout.

: la marionnette pivote d'un quart de tour à droite.

: la marionnette pivote d'un quart de tour à gauche.



: la marionnette agite les bras.



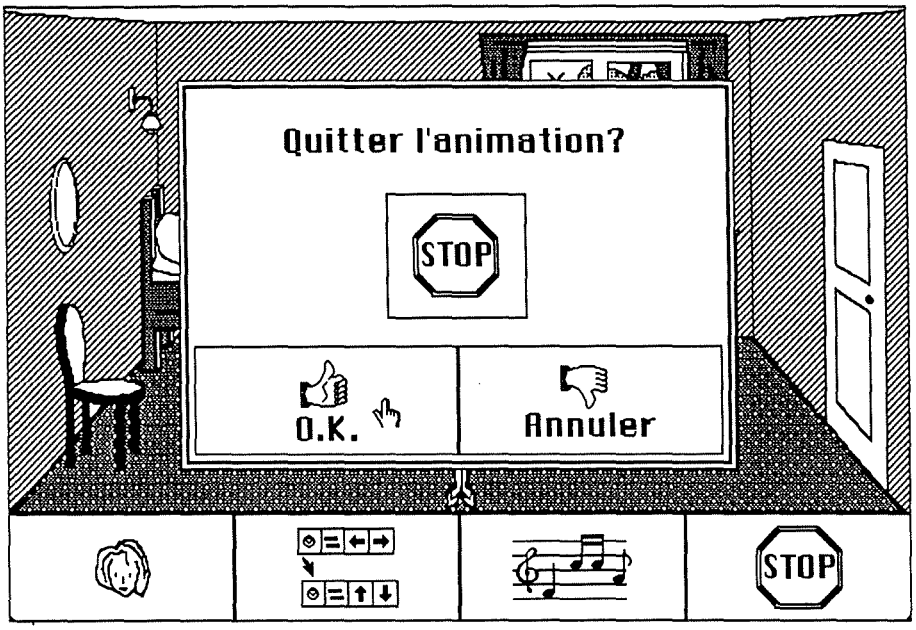
: arrêt ou mise en route de la musique.

: arrêt ou mise en route de l'enregistrement.

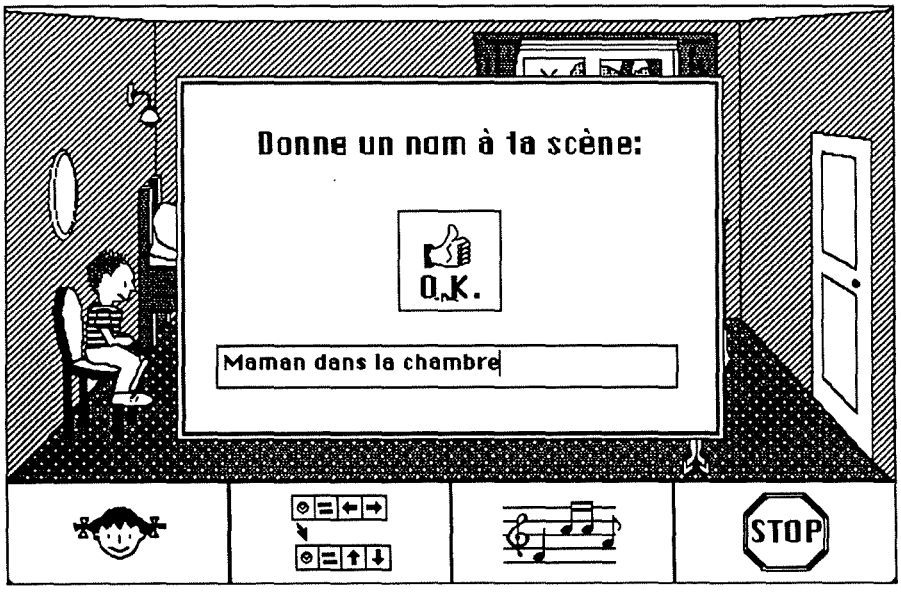
: sortie de la partie animation avec au préalable la fenêtre de confirmation (fenêtre 3). Si une scène a été enregistrée, la fenêtre 4 apparaît. Ensuite, retour à l'écran 3.

Pour les enfants qui utilisent le mode d'utilisation 'souris', il est possible de déplacer les personnages directement avec la souris plutôt que d'utiliser les quatre flèches de déplacement. Pour cela, il faut placer le curseur sur le personnage choisi et cliquer. Le curseur disparaît et lorsque l'on bouge la souris, le personnage la suit. Pour lâcher la marionnette, il suffit de cliquer une deuxième fois et le curseur réapparaît.

FENETRE 3:



FENETRE 4:

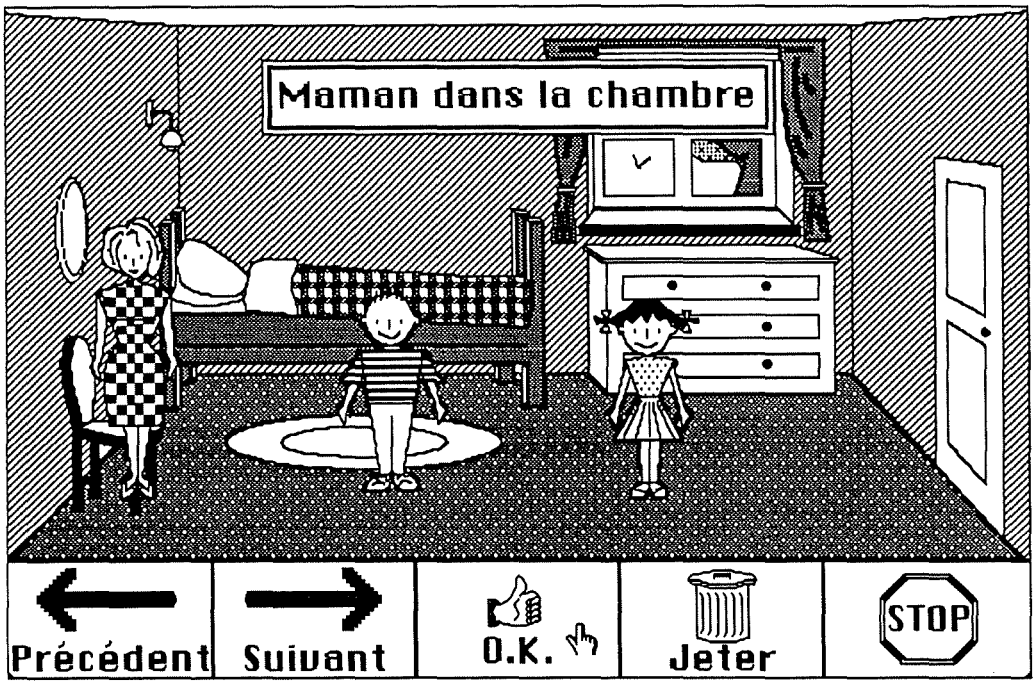


Lorsque l'enfant a confirmé sa commande STOP et qu'il a enregistré une scène, cette fenêtre apparaît. Le nom de la scène est introduit au clavier et est validé en cliquant sur l'icône 'stop' ou en appuyant sur <return>.

Option 'REGARDER UNE SCENE'.

Cette option permet d'aller visualiser les scènes qui ont été créées et enregistrées au préalable par l'enfant.

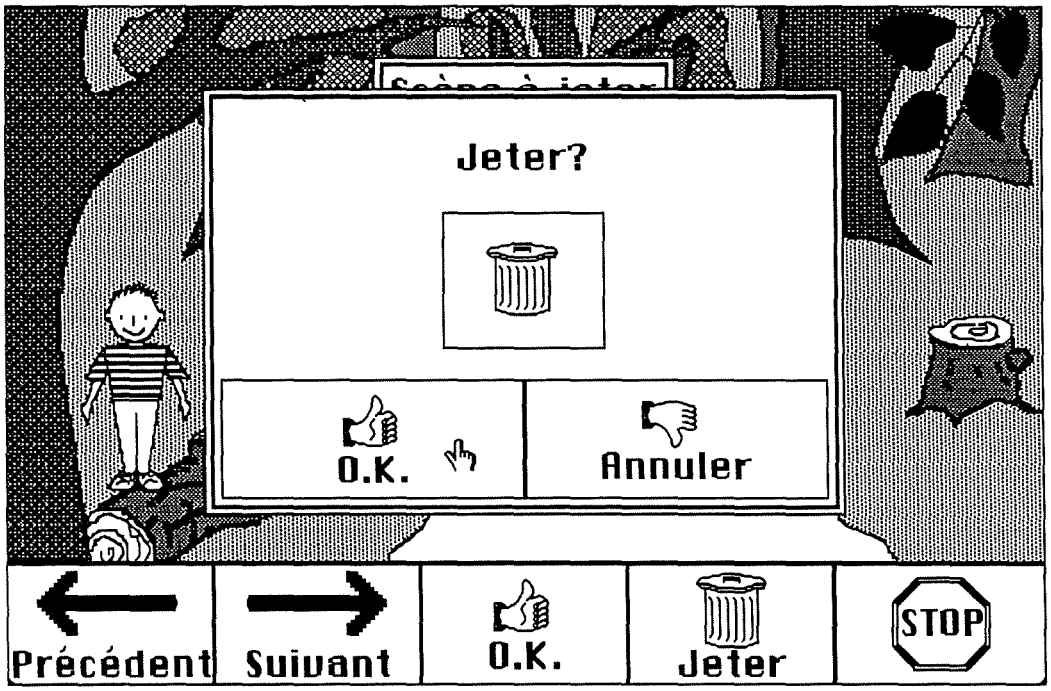
ECRAN 9.



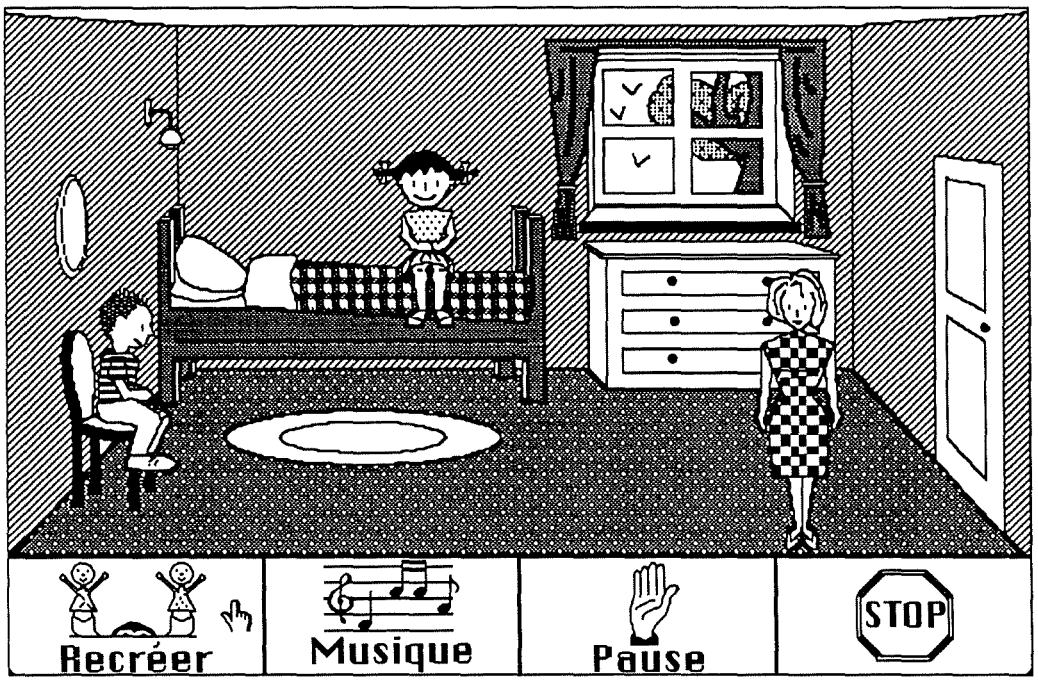
Cet écran présente les différentes scènes qu'il est possible de regarder. Les scènes sont présentées de la façon suivante : les marionnettes sont placées en ligne sur le décor et la musique se joue, le titre de la scène est affiché en haut de l'écran. Les commandes PRECEDENT et SUIVANT permettent de passer en revue les différentes scènes disponibles. Le déroulement est circulaire (idem que pour les décors). La commande JETER implique la suppression de la scène. Pour éviter des erreurs de manipulation et donc des suppressions de scènes que l'on voudrait conserver, une confirmation est demandée avant l'exécution de la commande. La commande STOP permet de revenir à l'écran 2 et ainsi de retourner dans l'autre option 'créer une scène'. Lorsque la scène que l'enfant veut regarder apparaît à l'écran, il valide la commande OK et la scène commence à se jouer.

Case 1 ---> passage à la scène précédente.
Case 2 ---> passage à la scène suivante.
Case 3 ---> passage à l'écran 10 où se joue la scène.
Case 4 ---> suppression de la scène (avec fenêtre 5 de confirmation).
Case 5 ---> retour à l'écran 2.

FENETRE 5.

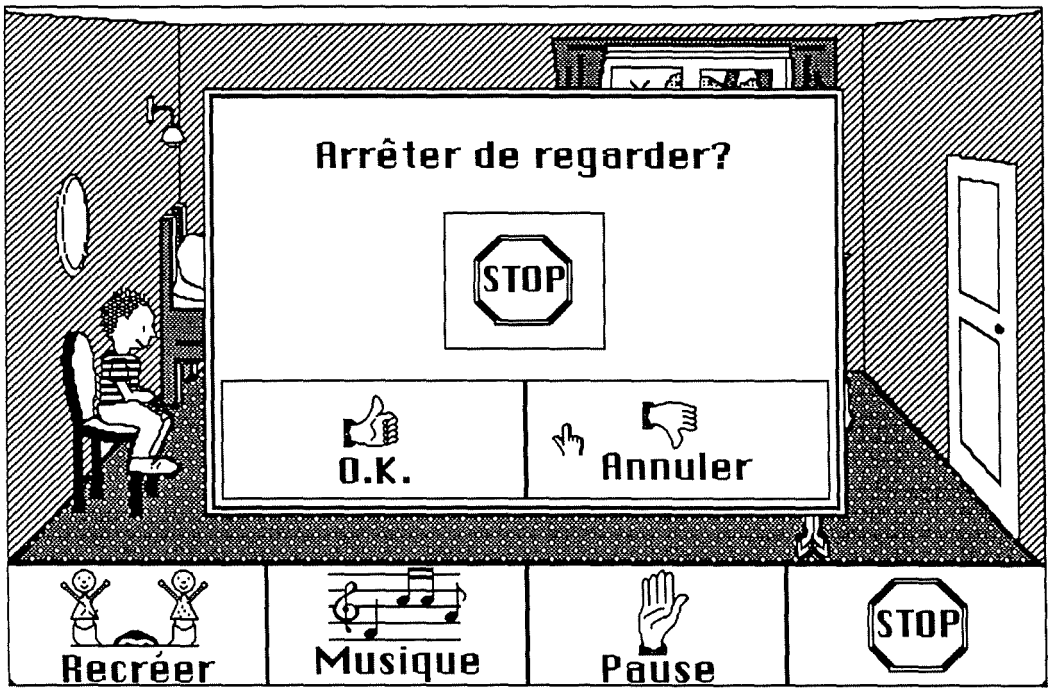


ECRAN 10.

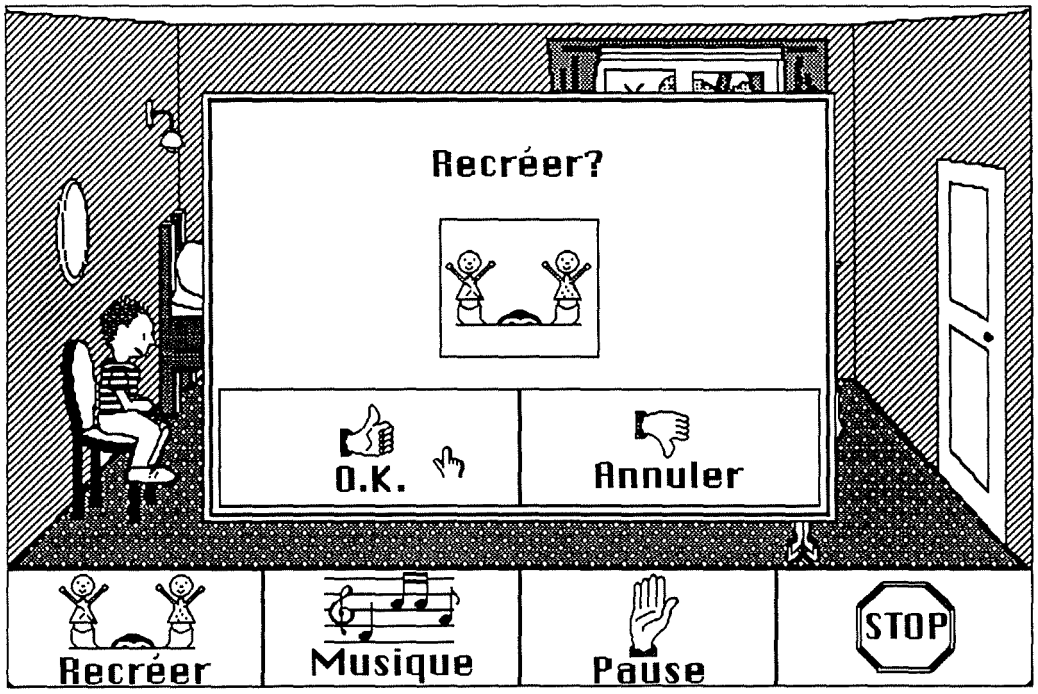


La scène se joue. Les commandes enregistrées s'exécutent les unes après les autres sans temps d'attente. Si l'enfant a attendu trois minutes entre deux commandes lors de l'enregistrement, celles-ci n'apparaissent pas lorsque la scène se rejoue. La commande PAUSE permet de faire un 'arrêt sur image'. Il faut de nouveau valider cette commande pour que la scène reparte. La commande MUSIQUE entraîne l'arrêt ou la mise en route de la musique. La commande STOP entraîne le retour à l'écran 9 après une demande de confirmation (fenêtre 6). OrdiThéâtre offre aussi la possibilité de continuer la scène enregistrée ou d'en modifier la fin. Lorsque l'on valide la commande RECREER, on passe directement (après une demande de confirmation, fenêtre 7) à la partie animation de l'option 'créer une scène'. Les personnages restent à la place où ils se trouvaient lorsque l'enfant a validé la commande 'recréer'. Si l'enfant branche la caméra, toute la partie de la scène qui n'avait pas encore été rejouée est effacée et ce qu'il enregistrera se placera directement derrière ce qui avait déjà été rejoué. De ce fait, si la scène avait été rejouée entièrement, on peut la continuer. Si elle avait été rejouée partiellement, on supprime la fin et on la recrée.

FENETRE 6.



FENETRE 7.



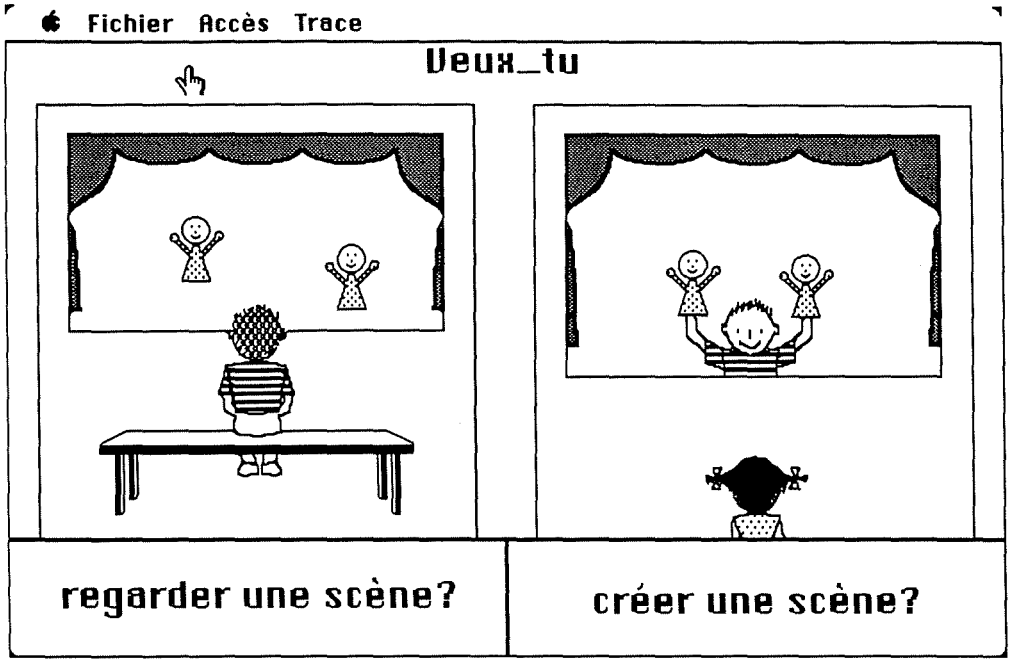
Toute la partie destinée à l'enfant a ainsi été présentée en détail. Tous les écrans ainsi que les commandes ont été explicités. Il ne vous reste plus qu'à jouer pour vous familiariser avec le logiciel ! Créez des scènes, regardez-les, modifiez-les, promenez-vous dans OrdiThéâtre, c'est le meilleur moyen de le maîtriser et ainsi, de pouvoir assister convenablement l'enfant lorsqu'il le découvrira. Lorsque vous le maîtrisez, vous pouvez passer à la partie destinée au moniteur et qui concerne la modification des différents paramètres présentés au point 1. Il s'agit du point 4. MENUS.

4. Les menus.

Le moniteur peut accéder à une barre de menus à partir de n'importe quel endroit du programme. C'est à l'aide de ces menus qu'il pourra modifier les différents paramètres expliqués au point 1. C'est aussi dans ces menus que se trouve l'option 'quitter' qui permet de quitter OrdiThéâtre.

Pour obtenir cette barre de menus, enfoncez simultanément la touche <commande> (⌘) et la touche <m>. Elle apparaît alors dans le haut de l'écran et elle comprend les menus suivants : , Fichier, Accès, Trace.

Pour faire disparaître cette barre de menus, il suffit d'enfoncer une deuxième fois simultanément la touche <commande> (⌘) et la touche <m>.



Lorsque la barre de menus est affichée en haut de l'écran, il faut utiliser la souris pour accéder aux diverses options disponibles dans les différents menus. Pour cela, placez le curseur de la souris sur le titre du menu auquel vous désirez accéder. Ensuite, enfoncez le bouton de la souris et maintenez-le dans cette position. Le menu se déroule et les options apparaissent. Pour sélectionner une option du menu, descendez le curseur de la souris dans les options en maintenant toujours le bouton enfoncé. Une bande noire apparaît, au fur et à mesure, sur l'option sur laquelle se trouve le curseur. Lorsque la bande noire se trouve sur l'option que vous désirez sélectionner, relâchez le bouton de la souris.

4.1. Le menu .

La première option de ce menu s'intitule 'A propos d'OrdiThéâtre'. Elle fournit des commentaires généraux concernant le logiciel.

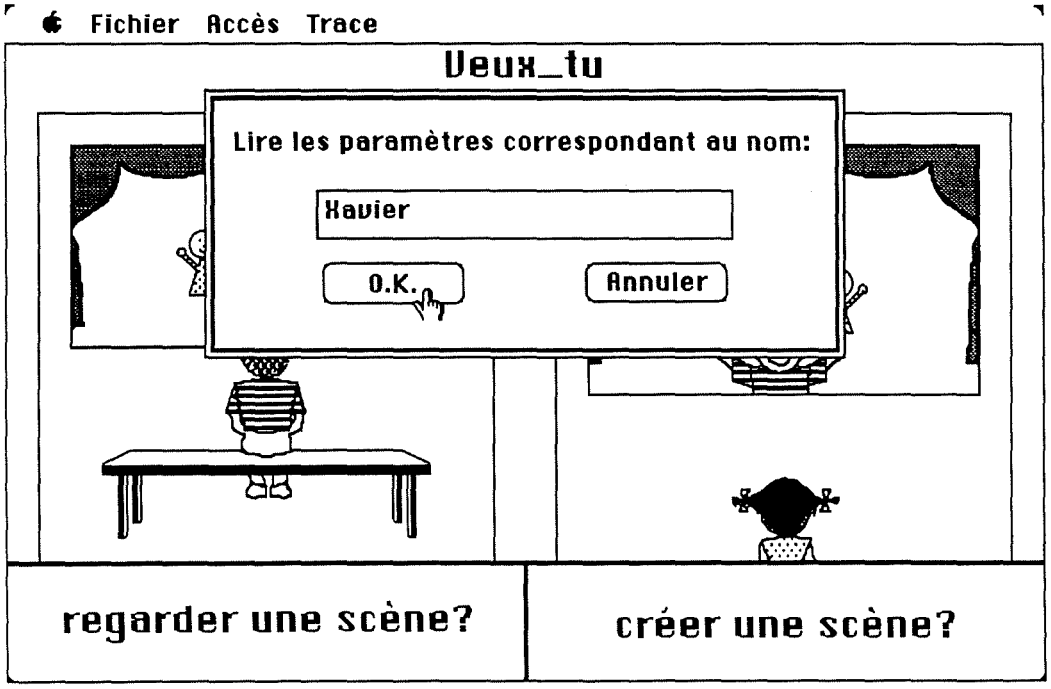
Les options suivantes sont des accessoires de bureau Macintosh. Ces accessoires ne sont pas indispensables pour l'utilisation d'OrdiThéâtre. C'est pourquoi ils ne sont pas détaillés dans ce manuel.

Pour obtenir des renseignements sur ces accessoires, se référer au manuel d'utilisation Macintosh.

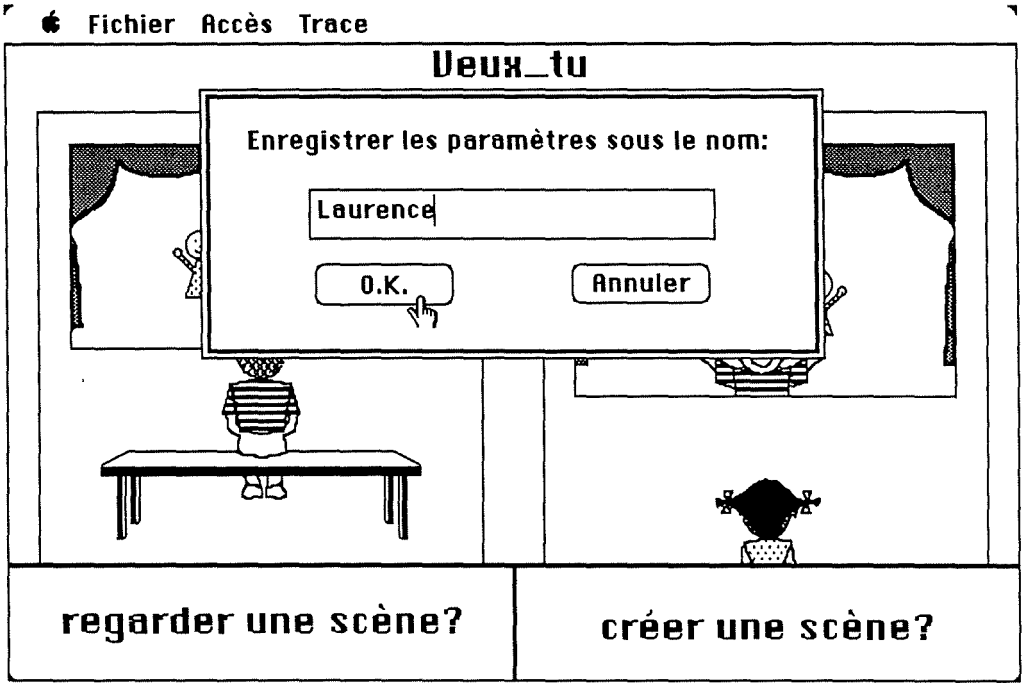
4.2. Le menu Fichier.

Les deux premières options concernent la lecture et l'enregistrement de nouveaux paramètres.

<Lire les paramètres> : permet d'aller chercher des paramètres enregistrés précédemment. Cette option est utile dans le cas où l'on change d'enfant en cours d'utilisation. En effet, le premier enfant, en entrant son nom à l'écran 1, a sélectionné les paramètres lui correspondants. Si l'on change d'enfant en cours de jeu, les paramètres ne seront plus adaptés. Il faut donc aller chercher les paramètres du deuxième enfant (pour autant que des paramètres aient été enregistrés sous son nom). Cela se fait en validant cette option 'lire les paramètres' au moyen de la souris. La fenêtre suivante apparaît et il suffit de rentrer le nom de l'enfant au clavier suivi de <return> ou d'un clic-souris dans la case <ok>. Les paramètres sont alors automatiquement modifiés et l'utilisation normale du logiciel peut reprendre son cours.

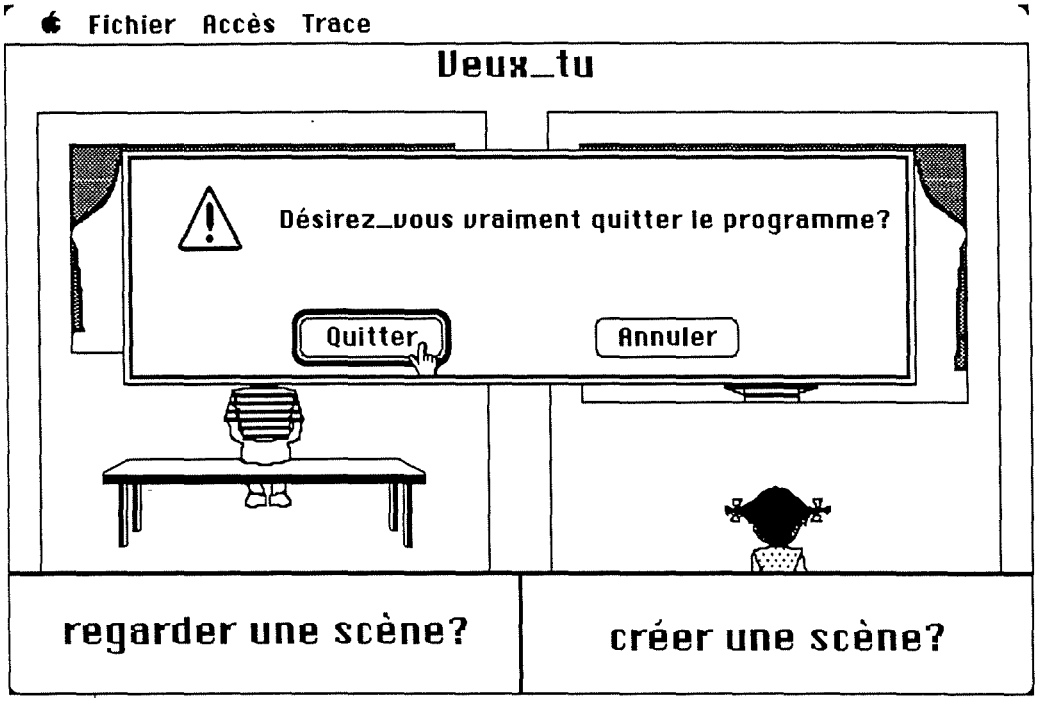


<Enregistrer les paramètres> : permet d'enregistrer de nouveaux paramètres pour un nouvel enfant . Lorsque des paramètres sont définis, on peut les enregistrer. OrdiThéâtre prendra donc automatiquement ces paramètres lorsque l'enfant rentrera son nom au début de la prochaine utilisation. Le nom s'introduit de la même manière que pour <lire les paramètres>.



<Enregistrement automatique> : cette option permet de brancher ou débrancher l'enregistrement automatique (expliqué au point 1.paramètres). Si le signe √ est présent à gauche de l'option, l'enregistrement automatique est branché, sinon il ne l'est pas. Il suffit de sélectionner l'option et l'enregistrement se branche s'il ne l'était pas et se débranche s'il était branché.

<Quitter> : permet de quitter le programme. Une confirmation est demandée au préalable.

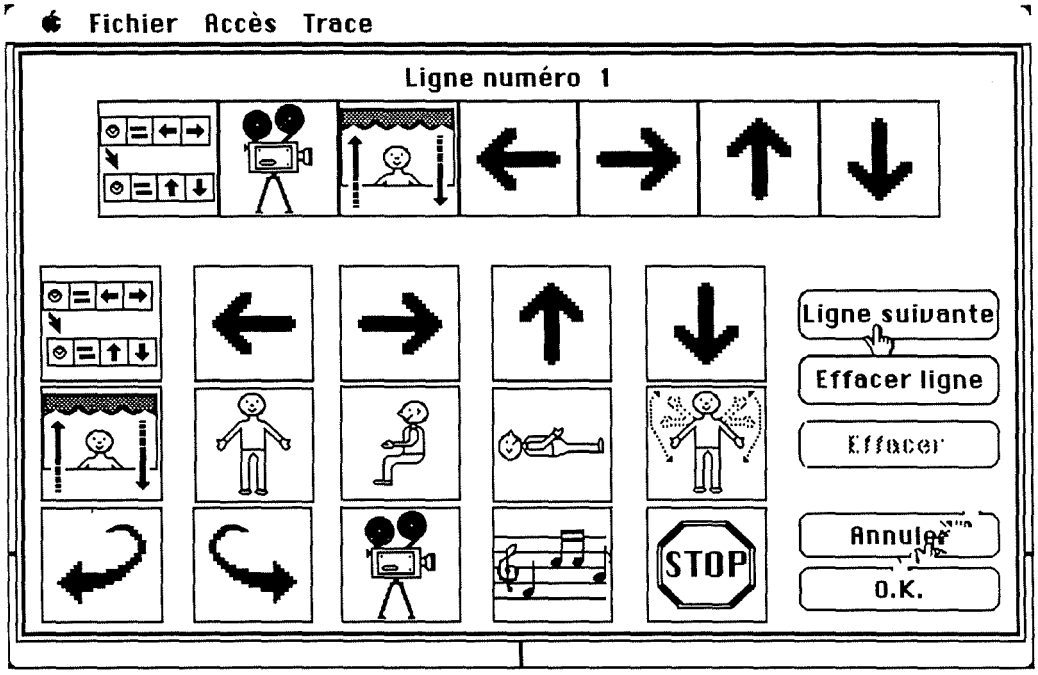


4.3. Le menu Accès.

Les trois premières options concernent le mode d'utilisation. Ces différents modes sont expliqués au point 1. paramètres. Le signe √ est affiché à gauche du mode d'utilisation sélectionné. Pour changer, il suffit de valider l'option choisie comme expliqué précédemment.

L'option suivante concerne la vitesse de défilement. Après avoir validé cette option, une fenêtre apparaît permettant de modifier la vitesse. La vitesse actuelle est affichée (c'est-à-dire le délai entre deux déplacements du curseur, en secondes) et un curseur se déplace dans le bas de cette fenêtre à cette vitesse. Si l'on clique sur la flèche de droite, le délai entre deux déplacements du curseur augmente et donc, la vitesse diminue. Si l'on clique sur la flèche de droite, c'est l'inverse. Le curseur du bas de la fenêtre permet de visualiser chaque vitesse possible. Lorsque l'on a trouvé la vitesse qui convient, on clique sur la case <ok>. La case <annuler> permet de rétablir la vitesse qui était en cours avant la sélection de cette option.

La dernière option de ce menu concerne l'organisation des lignes de commandes. Après validation de cette option, la fenêtre suivante apparaît :



Sur la première ligne apparaissent successivement les quatre lignes de commandes telles qu'elles sont définies actuellement. Pour les visualiser, il faut valider la case <LIGNE SUIVANTE> (après la quatrième ligne, on revient à la première).

En dessous, sont présentées les quinze commandes disponibles.

La case <EFFACER LIGNE> permet de supprimer,en une fois, toutes les commandes se trouvant sur la ligne affichée. Il est possible d'effacer une seule commande à la fois. Pour cela, cliquez sur la commande que vous désirez effacer. Celle-ci s'affiche en inverse. Ensuite, cliquez sur la commande <EFFACER>. La commande s'efface et laisse la case libre pour une autre commande.

Pour placer de nouvelles commandes sur la ligne, cliquez sur une des quinze commandes. Celle-ci s'affiche en inverse. Positionnez le curseur dans la ligne sur la case où vous désirez placer la commande. Cliquez et

la commande s'affiche dans cette case. Si la case n'était pas libre, la commande précédente est supprimée et est remplacée par la nouvelle.

Définissez ainsi vos quatre nouvelles lignes de commandes. Les lignes restées complètement vides n'apparaîtront pas lors de l'utilisation. De même, si certaines cases de commande ne sont pas utilisées, elles ne figureront pas sur la ligne à l'utilisation. La grandeur des cases utilisées sera adaptée de manière à remplir toute la ligne de commandes. Lorsque vous avez fini, validez la commande <OK>. Vous vous retrouvez à l'endroit où vous aviez choisi l'option <commandes>, et les lignes de commandes sont redéfinies. Si vous voulez revenir à cet endroit sans modifier les anciennes lignes, validez la commande <ANNULER>.

4.4. Le menu Trace.

Ce menu permet de branche la trace à différents moments de l'utilisation. Le signe √ est présent à gauche de l'option sélectionnée.

<aucune> : aucune trace ne sera enregistrée lors de l'utilisation.

<phase de sélection> : la trace sera enregistrée lors de la sélection des personnages, du décor et de la musique.

<phase d'animation> : la trace sera enregistrée lors de l'animation des marionnettes (écran 8).

<phase de sélection et d'animation> : la trace sera enregistrée lors des deux phases précédentes.

Lorsqu'une trace est demandée, elle est enregistrée dans un fichier séparé (il s'agit d'un fichier MacWrite). Celui-ci peut donc être facilement imprimé à l'aide de MacWrite ou d'un autre traitement de texte.

5. Commentaires.

Il est possible, à tout moment, d'effectuer une impression de l'écran sur papier. Cette impression ne comprend cependant pas la ligne de commandes du bas de l'écran, ni la barre de menus. Lorsqu'une imprimante est branchée, il suffit de presser simultanément les touches <commande> (⌘), <shift> (⇧) et <4>.

Les fichiers trace (fichiers de type MacWrite dont le nom se termine par ".trace.*numéro*") ne sont pas nécessaires au bon fonctionnement du programme. Ils peuvent donc être supprimés du dossier contenant l'application (disque dur) ou de la disquette "Disque OrdiThéâtre". Si vous désirez les conserver, il est conseillé de les stocker sur des disquettes différentes de celles de l'application, car ils peuvent prendre beaucoup de place!

ANNEXE 2 :

Questionnaire d'évaluation.

Procédure d'expérimentation.

STADE 1.

On place l'enfant directement devant l'écran d'animation. Celui-ci ne comprend pas de décor. Il n'y a que deux personnages disponibles (la petite fille et le petit garçon), placés au milieu de l'écran .
Deux lignes de commandes sont accessibles :
1:(tête du personnage)// chgt de ligne // entrée,sortie scène // gauche // droite // haut // bas.
2: " " // " " // debout // assis // couché // agiter // tourner // stop.

Attention, n'oubliez pas de brancher l'enregistrement automatique (voir menu 'fichier') avant d'accéder à l'écran d'animation.

Lorsque l'enfant maîtrise ces commandes, on passe au stade 2.

STADE 2.

On montre à l'enfant qu'il existe d'autres décors et d'autres personnages. On lui laisse la possibilité de choisir un décor et des personnages et ensuite de créer sa scène. De plus, on débranche l'enregistrement automatique et on ajoute la caméra sur la première ligne de commandes. A l'enfant de décider lui-même s'il veut enregistrer ce qu'il crée.

Lorsqu'il maîtrise les choix des personnages, du décor et l'animation, on passe au stade suivant.

STADE 3.

On place l'enfant devant l'écran de départ ('Veux-tu regarder...'). On lui explique qu'il peut regarder une scène qu'il a créée précédemment, qu'il peut la continuer ou en créer une nouvelle.

A compléter avant l'expérimentation.

* *Fiche signalétique de l'enfant.*

Nom :
Prénom :
Age :
Sexe :

Handicap moteur :

Handicap sensoriel :

Handicap mental (aucun, léger, modéré, sévère) :

Troubles associés :

Troubles du langage :

Acquis : - en lecture : lit son nom 0
des mots 0
des phrases courtes 0
les chiffres 0

- en écriture : écrit son nom 0
des mots 0
des phrases courtes 0
les chiffres 0

* *Familiarité avec l'ordinateur.*

L'enfant a-t-il déjà eu accès à un ordinateur ? jamais 0
rarement 0
souvent 0

Si oui, - contexte : divertissement 0
éducatif 0

* *Fiche signalétique du moniteur.*

Nom :
Prénom :
Fonction :

* *Objectifs poursuivis.*

A première vue, quels sont vos objectifs d'utilisation de ce logiciel avant sa mise en application ?

A. Du point de vue du savoir-faire (champ psycho-moteur) : motricité, perception,...

B. Du point de vue du savoir-être (champ socio-affectif) : affectivité, créativité, autonomie, communication, ...

C. Du point de vue ludique :

D. Autres :

A compléter pendant l'expérimentation.

Comment remplir le questionnaire ?

Ce questionnaire contient deux types de questions : des questions par écran et des questions générales.

Questions par écran :

Une première partie concerne la compréhension même des commandes. Il faut mettre le niveau de compréhension (1,2,3 ou 4) sous l'icone correspondant à la commande. Quatres niveaux de compréhension sont définis.

L'enfant comprend-il les commandes :

niveau 1 : oui, spontanément

niveau 2 : oui, après une explication minimale

niveau 3 : oui, avec une aide répétée

niveau 4 : non

La seconde partie se présente comme un questionnaire à choix multiples.

Questions générales :

Elles se présentent également comme un questionnaire à choix multiples.

SEANCE 1.

Date de la séance :
Heure du début de la séance :
Durée de la séance :
Stade d'expérimentation du logiciel (voir la procédure d'expérimentation) auquel l'enfant est arrivé en fin de séance :
Mode d'utilisation (+ aménagements nécessaires):

SEANCE 2.

Date de la séance :
Heure du début de la séance :
Durée de la séance :
Stade d'expérimentation du logiciel (voir la procédure d'expérimentation) auquel l'enfant est arrivé en fin de séance :
Mode d'utilisation (+ aménagements nécessaires):

SEANCE 3.

Date de la séance :
Heure du début de la séance :
Durée de la séance :
Stade d'expérimentation du logiciel (voir la procédure d'expérimentation) auquel l'enfant est arrivé en fin de séance :
Mode d'utilisation (+ aménagements nécessaires):

SEANCE 4.

Date de la séance :

Heure du début de la séance :

Durée de la séance :

Stade d'expérimentation du logiciel (voir la procédure d'expérimentation) auquel l'enfant est arrivé en fin de séance :

Mode d'utilisation (+ aménagements nécessaires):

SEANCE 5:

Date de la séance :

Heure du début de la séance :

Durée de la séance :

Stade d'expérimentation du logiciel (voir la procédure d'expérimentation) auquel l'enfant est arrivé en fin de séance :

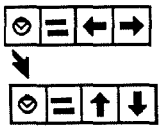
Mode d'utilisation (+ aménagements nécessaires):

Ecran d'animation.

Rappel : 1 : oui, spontanément
2 : oui, après une explication minimale
3 : oui, avec une aide répétée
4 : non

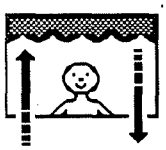
séance : 1 2 3 4 5

....



....

....



....

....

....

....

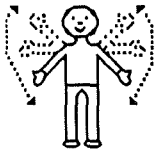
....

séance : 1 2 3 4 5

....

....

....



....

....

....

Ecran d'animation.

- Quelles sont les commandes qu'il utilise le plus souvent ?

séance 1 :

séance 2 :

séance 3 :

séance 4 :

séance 5 :

- Quelles sont les commandes qu'il utilise le moins souvent ?

séance 1 :

séance 2 :

séance 3 :

séance 4 :

séance 5 :

- la multiplicité des lignes de commandes perturbe-t-elle l'enfant ?

séance :	1	2	3	4	5
fortement	0	0	0	0	0
moyennement	0	0	0	0	0
pas du tout	0	0	0	0	0

- le décor étant une toile de fond, est-il troublé par le fait de ne pas pouvoir faire passer les personnages derrière les objets ?

séance :	1	2	3	4	5
fortement	0	0	0	0	0
moyennement	0	0	0	0	0
pas du tout	0	0	0	0	0

- cherche-t-il à déplacer les objets du décor ?

séance :	1	2	3	4	5
souvent	0	0	0	0	0
parfois	0	0	0	0	0
jamais	0	0	0	0	0

- s'il emploie la souris, utilise-t-il la possibilité de déplacer le personnage avec la souris ?

séance :	1	2	3	4	5
souvent	0	0	0	0	0
parfois	0	0	0	0	0
jamais	0	0	0	0	0

- donne-t-il un nom à la scène créée ?

séance :	1	2	3	4	5
souvent	0	0	0	0	0
parfois	0	0	0	0	0
jamais	0	0	0	0	0

Ecran des choix des personnages, du décor, de la musique.

Rappel : 1 : oui, spontanément 3 : oui, avec une aide répétée

2 : oui, après une explication minimale 4 : non

séance :	1	2	3	4	5
Décors	....	....	....	....	....
Personnages	....	....	....	....	....
Créer	....	....	....	....	....
Stop	....	....	....	....	....

L'enfant comprend-il

- qu'il doit sélectionner un décor, des personnages avant de créer une scène ?

séance :	1	2	3	4	5
oui, spontanément	0	0	0	0	0
oui, après une explication minimale	0	0	0	0	0
oui, avec une aide répétée	0	0	0	0	0
non	0	0	0	0	0

Ecran du choix du décor.

Rappel : 1 : oui, spontanément
2 : oui, après une explication minimale
3 : oui, avec une aide répétée
4 : non

séance : 1 2 3 4 5

Précédent ....

Suivant ....

O.K. ....

L'enfant

séance :	1	2	3	4	5
- passe-t-il tous les décors en revue avant de choisir ?	0	0	0	0	0
passe-t-il quelques décors en revue avant de choisir ?	0	0	0	0	0
choisit-il le premier décor qui lui est présenté ?	0	0	0	0	0

- quels décors choisit-il ?

séance 1 :
séance 2 :
séance 3 :
séance 4 :
séance 5 :

Ecran du choix des personnages.

Rappel : 1 : oui, spontanément
2 : oui, après une explication minimale
3 : oui, avec une aide répétée
4 : non

séance : 1 2 3 4 5

Précédent ....

Suivant ....

Prendre ....

Modifier ....

O.K. ....

L'enfant

séance :	1	2	3	4	5
- passe-t-il tous les personnages en revue avant de choisir ?	0	0	0	0	0
passe-t-il quelques personnages en revue avant de choisir?	0	0	0	0	0
choisit-il le premier personnage présenté à l'écran ?	0	0	0	0	0

- comprend-il qu'il peut choisir entre 1 et 4 personnages ?

séance :	1	2	3	4	5
oui, spontanément	0	0	0	0	0
oui, après une explication minimale	0	0	0	0	0
oui, avec une aide répétée	0	0	0	0	0
non	0	0	0	0	0

- utilise-t-il la possibilité de modifier son choix ?

séance :	1	2	3	4	5
souvent	0	0	0	0	0
parfois	0	0	0	0	0
jamais	0	0	0	0	0

- quels personnages choisit-il ?

séance 1 :

séance 2 :

séance 3 :

séance 4 :

séance 5 :

Ecran de modification du choix des personnages.

Rappel : 1 : oui, spontanément
2 : oui, après une explication minimale
3 : oui, avec une aide répétée
4 : non

séance : 1 2 3 4 5

....

....

L'enfant

- utilise-t-il la possibilité d'effacer plusieurs personnages ?

séance :	1	2	3	4	5
souvent	0	0	0	0	0
parfois	0	0	0	0	0
jamais	0	0	0	0	0

-Ecran de départ .

Rappel : 1 : oui, spontanément 3 : oui, avec une aide répétée

2 : oui, après une explication minimale 4 : non

séance :	1	2	3	4	5
regarder une scène?	....	....	....	....	....
créer une scène?	....	....	....	....	....

Ecran de choix de la scène à regarder.

Rappel : 1 : oui, spontanément
2 : oui, après une explication minimale
3 : oui, avec une aide répétée
4 : non

séance :	1	2	3	4	5
 Précédent	....	....	....	....	....
 Suivant	....	....	....	....	....
 O.K.	....	....	....	....	....
 Jeter	....	....	....	....	....
	....	....	....	....	....

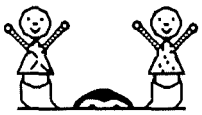
L'enfant

séance :	1	2	3	4	5
- passe-t-il les scènes en revue avant de choisir ?	0	0	0	0	0
prend-il la première qui lui est présentée ?	0	0	0	0	0

Ecran de visualisation de la scène.

Rappel : 1 : oui, spontanément 3 : oui, avec une aide répétée
2 : oui, après une explication minimale 4 : non

séance : 1 2 3 4 5



Recréer ....

Pause ....

....

L'enfant

- utilise-t-il la pause ?

séance :	1	2	3	4	5
souvent	0	0	0	0	0
parfois	0	0	0	0	0
jamais	0	0	0	0	0

- utilise-t-il la possibilité de recréer la scène ?

séance :	1	2	3	4	5
souvent	0	0	0	0	0
parfois	0	0	0	0	0
jamais	0	0	0	0	0

Commentaires :

Questions générales.

L'enfant

- comprend-il qu'il doit toujours utiliser la ligne de commandes du bas de l'écran ?

séance :	1	2	3	4	5
oui, spontanément	0	0	0	0	0
oui, après une explication minimale	0	0	0	0	0
oui, avec une aide répétée	0	0	0	0	0
non	0	0	0	0	0

- se base-t-il sur les dessins pour se retrouver ?

séance :	1	2	3	4	5
uniquement	0	0	0	0	0
en partie	0	0	0	0	0
pas du tout	0	0	0	0	0

- se base-t-il sur le texte pour se retrouver ?

séance :	1	2	3	4	5
uniquement	0	0	0	0	0
en partie	0	0	0	0	0
pas du tout	0	0	0	0	0

- clique-t-il sur des objets non interactifs ?

séance :	1	2	3	4	5
souvent	0	0	0	0	0
parfois	0	0	0	0	0
jamais	0	0	0	0	0

- renonce-t-il à sa commande suite à la demande de confirmation ?

séance :	1	2	3	4	5
souvent	0	0	0	0	0
parfois	0	0	0	0	0
jamais	0	0	0	0	0

- si l'enfant utilise la souris, place-t-il mal le curseur de la souris à cause de la forme de celui-ci ?

séance :	1	2	3	4	5
souvent	0	0	0	0	0
parfois	0	0	0	0	0
jamais	0	0	0	0	0

- le temps de réaction du logiciel à certains moments perturbe-t-il son emploi ?

séance :	1	2	3	4	5
fortement	0	0	0	0	0
moyennement	0	0	0	0	0
pas du tout	0	0	0	0	0

- utilise-t-il la commande 'précédent' ?

séance :	1	2	3	4	5
souvent	0	0	0	0	0
parfois	0	0	0	0	0
jamais	0	0	0	0	0

- utilise-t-il la commande 'suivant' ?

séance :	1	2	3	4	5
souvent	0	0	0	0	0
parfois	0	0	0	0	0
jamais	0	0	0	0	0

- regarde-t-il une scène créée lors d'une séance précédente ?

séance :	1	2	3	4	5
souvent	0	0	0	0	0
parfois	0	0	0	0	0
jamais	0	0	0	0	0

- la séance est interrompue

séance :	1	2	3	4	5
par l'enfant ?	0	0	0	0	0
l'adulte ?	0	0	0	0	0

- pour quelles raisons :

séance 1 :

séance 2 :

séance 3 :

séance 4 :

séance 5 :

Réactions de l'enfant.

En général, fait-il des commentaires ?

Lesquels ?

A-t-il manifesté du plaisir ?

séance :	1	2	3	4	5
toujours	0	0	0	0	0
souvent	0	0	0	0	0
parfois	0	0	0	0	0
jamais	0	0	0	0	0

A-t-il manifesté de la lassitude ?

séance :	1	2	3	4	5
toujours	0	0	0	0	0
souvent	0	0	0	0	0
parfois	0	0	0	0	0
jamais	0	0	0	0	0

A-t-il manifesté de l'insatisfaction ?

séance :	1	2	3	4	5
toujours	0	0	0	0	0
souvent	0	0	0	0	0
parfois	0	0	0	0	0
jamais	0	0	0	0	0

A-t-il demandé à réutiliser le logiciel ?

séance :	1	2	3	4	5
toujours	0	0	0	0	0
souvent	0	0	0	0	0
parfois	0	0	0	0	0
jamais	0	0	0	0	0

Commentaires :

* *Variations des objectifs après utilisation.*

Quels sont vos objectifs de départ qui ont été satisfaits ?

Percevez-vous de nouveaux objectifs?

A compléter après l'ensemble des séances d'expérimentation.

* *Fiche signalétique du moniteur.*

Nom :
Prénom :
Fonction :

Nombre d'enfants vus :

Nombre de séances par enfant :

* *Critique du logiciel.*

A. Sur le plan technique :

- possibilité de réglage des paramètres :

- facilité d'utilisation (mise en route, menus, choix, adaptaion à divers handicaps, possibilité d'annuler en cas d'erreur,...) :

- qualité du graphisme (notamment la dimension des caractères et des dessins), de l'animation, {du son}; problèmes liés à la taille de l'écran, au noir et blanc, ... :

- autres remarques :

B. Sur le plan pédagogique et thérapeutique :

- utilité, comparaison avec le(s) système(s) utilisé(s) jusqu'alors :

- adaptabilité à différents types d'utilisateurs :

* *Propositions d'améliorations :*

* *Critères d'utilisation.*

Critères jugés indispensables :
- sur le plan moteur :

- sur le plan sensoriel :

- sur le plan du développement intellectuel :

- autres :

ANNEXE 3 :

Résultats de l'évaluation.

1. Tableaux des résultats.

Résultats de l'évaluation.

Onze enfants âgés de 5 à 11 ans ont participé à l'évaluation. Les réponses des questionnaires sont regroupées dans les tableaux suivants et sont présentées suivant l'ordre des questions.

Rappel : niveau 1 = oui, spontanément.
niveau 2 = oui, après une explication minimale.
niveau 3 = oui, avec une aide répétée.
niveau 4 = non.
A = abstention.

Ecran d'animation : compréhension des commandes.

Séances	1					2					3					4					5				
Niveau	1	2	3	4	A	1	2	3	4	A	1	2	3	4	A	1	2	3	4	A	1	2	3	4	A
chgt pers		7	3	1		6	2	3			10	1				6	2			3	8				3
chgt ligne		5	4	2		5	2	3	1		9	1		1		5	2		1	3	6	1	1		3
caméra		5	2	3	1	5	1	1	3	1	6	1	1	1	2	4	1		2	4	4	1		2	4
entr./sort.		5	5	1		7	1	2	1		8	2			1	5	2	1		3	8				3
gauche		9	2			8	3				9	2				6	1	1		3	7		1		3
droite		9	2			8	3				9	2				6	1	1		3	6	1	1		3
haut		9	2			7	2			2	8	2			1	5	1	1		4	6	2			3
bas		9	2			7	2			2	8	2			1	5	1	1		4	6	1	1		3
debout		8		1	2	5	2			4	8				3	5	1			5	4	1			6
assis	1	9		1		9	1			1	11					7				4	7				4
couché		10		1		10	1				11					8				3	6				5
agiter	1	9		1		9				2	11					8				3	7				4
tourner		6	3	1	1	5	2	1		3	6	3	1		1	4		1	1	5	3	1	1	1	5
stop	1	1	3	5	1	3	3	1	3	1	4	3	1	2	1	2	3	1	1	4	4	1	1	1	4

Ecran d'animation : commandes utilisées le plus et le moins souvent.

Séances	1		2		3		4		5	
Fréquence	+	-	+	-	+	-	+	-	+	-
chgt pers	3	1	6	1	6		4		5	
chgt ligne	2	4	5	2	5	1	2	2	3	2
caméra	1	7	2	5	1	5	2	2	3	4
entr./sort.	1	4	3	3	2	2	4		3	
gauche	8		7		7		6	1	6	1
droite	8	1	7		6		7	1	6	1
haut	5	2	9		8		7	1	6	1
bas	5	2	7		6		7	1	6	1
debout	1	3	1	5	2	3		3	1	4
assis	7	1	6	2	4	2		3	2	2
couché	4	1	2	2		1	1	2	1	3
agiter	6	1	5	3	4	2	3	2	4	3
tourner		4	2	6	2	4		6	2	3
stop		6		7	1	7		3		5

Ecran d'animation : mult. = multiplicité des lignes de commandes
décor = le décor est une toile de fond
dépl. obj. = déplacer les objets du décor
dépl.souris = déplacer les personnages à la souris
nom scène = donner un nom à la scène créée

F = fortement M = moyennement P = pas du tout
S = souvent P = parfois J = jamais
A = abstention

Séances	1				2				3				4				5			
	F	M	P	A	F	M	P	A	F	M	P	A	F	M	P	A	F	M	P	A
mult.	4	3	4		2	5	4		2	4	5		2	3	3	3	2	3	3	3
décor		1	10			1	10				11				8	3		1	7	3
	S	P	J	A	S	P	J	A	S	P	J	A	S	P	J	A	S	P	J	A
dépl. obj.		1	10			2	9				11				8	3		1	7	3
dépl.souris	2	4	4	1	4	4	2	1	3	4	3	1	2		5	4	4		3	4
nom scène	3	1	7		2	3	6		4	3	4		3	1	4	3	5		3	3

Ecran de choix des personnages, du décor, de la musique.

sél.obj. = sélection des objets avant de créer

Séances	1	2	3	4	5
Niveau	1 2 3 4 A	1 2 3 4 A	1 2 3 4 A	1 2 3 4 A	1 2 3 4 A
décors	5 2 1 3	3 2 4 2	5 3 2 1	3 3 1 4	5 2 4
pers.	5 2 1 3	3 2 4 2	5 3 2 1	3 3 1 4	5 1 1 4
créer	3 2 3 3	2 3 2 2 2	3 3 3 1 1	1 4 2 4	1 5 1 4
stop	5 2 2 2	4 2 3 2	1 5 1 2 2	1 4 2 4	5 2 4
sél. obj.	5 2 2 2	3 3 4 1	5 3 3	3 3 1 4	4 3 4

Ecran de choix du décor.

T = tous les décors Q = quelques décors P = premier décor

Séances	1	2	3	4	5
Niveau	1 2 3 4 A	1 2 3 4 A	1 2 3 4 A	1 2 3 4 A	1 2 3 4 A
précédent	3 2 3 3	1 1 3 2 4	2 1 4 2 2	2 2 2 5	2 1 2 1 5
suivant	1 4 2 1 3	3 3 2 3	6 2 1 1 1	3 2 1 1 4	5 2 4
ok	3 3 2 3	2 3 2 1 3	4 3 1 2 1	2 2 3 4	5 1 1 4
	T Q P A	T Q P A	T Q P A	T Q P A	T Q P A
choix déc.	7 1 1 2	7 3 1	7 4	5 1 1 4	5 2 4
forêt	9	6	9	2	4
s-à-mang.	4	7	5	3	2
salon	4	8	5	5	5

Ecran de choix des personnages.

T = tous les personnages Q = quelques personnages
P = premier personnage A = abstention

S = souvent P = parfois J = jamais A = abstention

Séances	1	2	3	4	5
Niveau	1 2 3 4 A	1 2 3 4 A	1 2 3 4 A	1 2 3 4 A	1 2 3 4 A
précédent	2 2 3 4	1 1 1 5 3	2 2 2 3 2	1 2 2 6	1 1 2 1 6
suivant	7 1 3	3 5 1 2	7 1 1 1 1	3 1 1 1 5	4 1 2 4
prendre	4 4 3	2 3 4 2	5 3 2 1	2 2 2 5	5 2 4
modifier	1 3 3 4	3 2 1 5	5 1 1 1 3	1 1 1 8	3 1 1 1 5
ok	1 4 2 4	2 1 3 2 3	5 1 1 2 2	2 1 2 6	3 3 1 4
1 à 4 pers.	4 3 1 1 2	7 1 2 1	7 1 1 1 1	6 2 3	4 1 1 5
	T Q P A	T Q P A	T Q P A	T Q P A	T Q P A
choix pers	4 5 2	4 6 1	3 6 1 1	2 5 4	7 4
	S P J A	S P J A	S P J A	S P J A	S P J A
modifier	1 3 5 2	1 3 6 1	6 5	3 5 3	3 2 6
maman	8	8	7	5	6
papa	6	9	7	6	8
fille	10	9	8	5	10
garçon	9	7	8	3	6
loup	6	7	11	4	6
chien	4	8	5	3	1
fauteuil rl.			5	4	6

Ecran de modification du choix des personnages.

S = souvent P = parfois J = jamais A = abstention
eff.plus.pe = effacer plusieurs personnages

Séances	1	2	3	4	5
Niveau	1 2 3 4 A	1 2 3 4 A	1 2 3 4 A	1 2 3 4 A	1 2 3 4 A
ok	3 1 3 4	3 2 6	4 2 2 2 1	1 2 8	2 3 1 5
effacer	2 2 3 4	2 1 2 6	5 1 2 1 2	1 1 1 8	2 2 1 1 5
	S P J A	S P J A	S P J A	S P J A	S P J A
eff.plus.pe	1 4 4 2	1 2 6 2	7 3 1	2 5 4	1 3 2 5

Questions générales.

U = uniquement P = en partie N = non, pas du tout A = abstention

S = souvent P = parfois J = jamais A = abstention

F = fortement M = moyennement P = pas du tout A = abstention

E = enfant Ad = adulte A = abstention

T = toujours S = souvent P= parfois J = jamais
A = abstention

Séances	1					2					3					4					5				
Niveau	1	2	3	4	A	1	2	3	4	A	1	2	3	4	A	1	2	3	4	A	1	2	3	4	A
ligne com.	3	3	4		1	5	4	1		1	6	4			1	7	1			3	7				4
	U	P	N	A		U	P	N	A		U	P	N	A		U	P	N	A		U	P	N	A	
dessins	8	3				8	3				8	3				8	1		2		6	2		3	
texte		3	8				3	8				3	8				1	8	2			2	6	3	
	S	P	J	A		S	P	J	A		S	P	J	A		S	P	J	A		S	P	J	A	
obj.non int			10	1		1		9	1			1	9	1				8	3			1	6	4	
dem. conf.		1	7	3				8	3				8	3			2	4	5			2	3	6	
curseur	1	1	7	2		1	1	7	2			2	6	3			2	5	4			2	3	6	
précédent	2	2	4	3		1	1	6	3		1	4	3	3			4	3	4			2	3	6	
suivant	6	2		3		8		1	2		6	3	1	1		4	3		4		4	1		6	
sc. créée			6	5		2	2	5	2		4	2	4	1		3	2	4	2		3	2	3	3	
	F	M	P	A		F	M	P	A		F	M	P	A		F	M	P	A		F	M	P	A	
temps réac	1	2	7	1			4	6	1		1		9	1				8	3			2	5	4	
	E		Ad		A	E		Ad		A	E		Ad		A	E		Ad		A	E		Ad		A
sc. interr.	1		10			5		6			9		2			4		5		2	3		4		4
	T	S	P	J	A	T	S	P	J	A	T	S	P	J	A	T	S	P	J	A	T	S	P	J	A
plaisir	3	6	1		1	3	6	2			1	7	3			3	3	1	1	3	2	5	1		3
lassitude			3	7	1			5	5	1			8	3				4	5	2			3	5	3
insatisfact.			3	8				3	8				5	6			1		8	2				8	3
réutiliser	7	2	1	1		6	3	1	1		4	2	2	3		4	2	2	1	2	4	2	2		3

2. Commentaires des utilisateurs.

A compléter avant l'expérimentation.

* *Fiche signalétique de l'enfant.*

Nom : ZINAÏ
Prénom : Marie Laure
Age : 5 ans
Sexe : F

Handicap moteur :

maladie de LiHe - ~~marche~~ se déplace avec ~~déambulateur~~ une canne.

Handicap sensoriel :

NON

Handicap mental (aucun, léger, modéré, sévère) :

niveau verbal et de raisonnement moyens.
trouble de représentation de l'espace

Troubles associés :

non

Troubles du langage :

non.

* *Fiche signalétique du moniteur.*

Nom : LATTY + de BARBOT
Prénom : Lauren Françoise
Fonction : stagiaire psych. psychologue

* *Objectifs poursuivis.*

A première vue, quels sont vos objectifs d'utilisation de ce logiciel avant sa mise en application ?

A. Du point de vue du savoir-faire (champ psycho-moteur) : motricité, perception,...

B. Du point de vue du savoir-être (champ socio-affectif) : affectivité, créativité, autonomie, communication, ...

C. Du point de vue ludique :

D. Autres : Dans cette étape :
- évaluation du logiciel {technique + intérêt.
- observation de l'enfant.

Commentaires : Marie Laure a eu au début beaucoup de mal à comprendre les commandes, d'un point, et à les manipuler, en raison de son handicap = 1) a du mal à comprendre qu'elle doit cliquer uniquement à droit et qu'elle ne doit pas toucher à la souris entre le moment où elle l'a positionnée et le "clic". 2) elle a du mal à effectuer ses gestes en raison de son handicap. Elle avance doucement, et a du mal à faire évoluer les personnages comme elle le veut sur l'écran.

Cependant, ça ne l'a pas empêché d'avoir une activité très riche pendant 6 séances.

1) elle s'identifie fortement à la petite fille, et exprime à travers elle ses difficultés d'enfant handicapée : l'impuissance motrice, notamment. Mais aussi l'angoisse qu'elle a eu elle comme tout le monde et que revoir a beaucoup impressionnant.

2) elle a beaucoup aimé ce jeu qui lui permet non seulement de dire difficultés et angoisse, mais aussi de les traiter, de les soigner.

Il faut noter l'utilisation importante qu'a fait Marie Laure des scènes existantes que l'on peut regarder, et à partir desquelles aussi l'on peut créer. "J'aime mieux regarder que créer", a-t-elle dit plusieurs fois.

* *Variations des objectifs après utilisation.*

Quels sont vos objectifs de départ qui ont été satisfaits ?

Il s'agissait d'une évaluation du logiciel...

Ces 6 séances ont permis une observation particulièrement intéressante de cet enfant qui, habituellement, contrôle son angoisse par un langage très adapté et n'exprime pas ses "problèmes" (problèmes familiaux en particulier). Or ceux-ci ont été évoqués dans ce contexte privilégié... L'outil "ardoise magique" a donc une grande force pour susciter les projections...

Percevez-vous de nouveaux objectifs?

Je pense que pour elle, il y aurait une utilisation psychothérapeutique particulièrement intéressante.

A compléter avant l'expérimentation.

* *Fiche signalétique de l'enfant.*

Nom : MOUA
Prénom : Nay Tsu
Age : 11 ans dans quelques jours.
Sexe : F

Handicap moteur :

non

Handicap sensoriel :

non

Handicap mental (aucun, léger, modéré, sévère) :

Léger.

Troubles associés :

Troubles du langage : Dysphasie avec difficultés de compréhension verbale et en expression très difficile et limité

* *Fiche signalétique du moniteur.*

Nom : Latty
Prénom : Laurence
Fonction : Psychologue stagiaire

+ de BARBOT
Françoise
Psychologue

* *Objectifs poursuivis.*

A première vue, quels sont vos objectifs d'utilisation de ce logiciel avant sa mise en application ?

A. Du point de vue du savoir-faire (champ psycho-moteur) : motricité, perception,...

B. Du point de vue du savoir-être (champ socio-affectif) : affectivité, créativité, autonomie, communication, ...

Moyen d'expression me paraissant privilégié pour ces enfants, dans le domaine de l'affectivité et de la créativité.

C. Du point de vue ludique :

D. Autres : ... peut-être pour certains, pour une initiation particulièrement attractive et riche d'expérimentation possible pour un enfant, le déclenchement de motivations d'ordre professionnel.

Dans cette étape 4 l'objectif ~~autre~~ premier est l'évaluation du logiciel (technique et intérêt

Commentaires : May Tsu manifeste par le sourire en fin de séance et entre 2 séances, qu'elle prend plaisir à ce logiciel.

Elle a compris très vite les commandes, et a joué rapidement avec tous les personnages.

Si au début elle a utilisé surtout les enfants et les animaux dans un jeu très ludique, ensuite elle a beaucoup joué des scènes de famille, où elle figure toujours (petite fille), et aussi sa maman, souvent sa petite sœur et des enfants du service qu'elle aime bien.

La question s'est posée de savoir si elle n'était pas un peu grande pour s'identifier à une petite fille, comme sa propre sœur qui a 4 mois. Elle m'a répondu d'un sourire que ce n'était pas le cas.

Elle a fait des scènes relativement courtes, où elle a toujours pris plaisir.

J'y ai vu la même enfant sereine et calme, la vie harmonieuse qui se reflète dans son regard, que dans les couloirs ou dans sa classe.

Concernant les titres qu'elle donne aux scènes, on y voit une nette évolution dans le sens de la confia

1°) rien
2°) "May Tsu"
3°) "May Tsu, Juliette, Laurence, et Denis".

* *Variations des objectifs après utilisation.*

Quels sont vos objectifs de départ qui ont été satisfaits ?

Observation intéressante puisque May Tsou ne peut s'exprimer par le langage = objectif atteint, puisqu'on a pu confirmer l'impression de sérénité qu'elle donne.—
Le logiciel a, dans ce cas - eu valeur diagnostique..

Percevez-vous de nouveaux objectifs?

Une utilisation pour développer la créativité de May Tsou serait intéressante - mais l'outil me semble un peu limité pour un enfant de cet âge.

A compléter avant l'expérimentation.

* *Fiche signalétique de l'enfant.*

Nom : VAUTROT
Prénom : Julie
Age : 6a
Sexe : F

Handicap moteur :
athétose majeure. pas de déambulation ni de manipulation.

Handicap sensoriel :
non.

Handicap mental (aucun, léger, modéré, sévère) :
niveau subnormal

Troubles associés :
non.

Troubles du langage :
Parole impossible en raison du handicap moteur.

* *Fiche signalétique du moniteur.*

Nom : LATTY + de BARBOT
Prénom : Laurence Françoise
Fonction : stagiaire psych. psychologue.

* *Objectifs poursuivis.*

A première vue, quels sont vos objectifs d'utilisation de ce logiciel avant sa mise en application ?

A. Du point de vue du savoir-faire (champ psycho-moteur) : motricité, perception,...

B. Du point de vue du savoir-être (champ socio-affectif) : affectivité, créativité, autonomie, communication, ...

C. Du point de vue ludique :

D. Autres : Dans cette étape :
- évaluation du logiciel sur le plan accessibilité et intérêt

Commentaires : Lulu n. s'est intéressé qu'au(x) loup(s). A la dernière séance j'ai essayé de savoir pourquoi, et ai fait référence à d'autres jeux auxquels elle a joué dans le service. Elle a reconnu qu'elle aime les loups parce qu'ils font ce qu'ils veulent, et elle pas. Puis elle a redemandé deux personnages : 2 femmes, qu'elle a fait entrer sans dire mais n'en a pas fait grand chose. Puis elle a eu l'air fatigué, et on a arrêté.

* *Variations des objectifs après utilisation.*

Quels sont vos objectifs de départ qui ont été satisfaits ?

On a montré que l'outil était accessible à cet enfant à handicap extrême - (avec des aménagements)
On a montré qu'elle a projeté dans les scènes créées des éléments de sa problématique psych. peu exprimables autrement - (en raison du handi. de communication)

Percevez-vous de nouveaux objectifs?

Utilisation psychothérapeutique possible - dans le sens d'une occasion pour elle d'exprimer ce qui la préoccupe -, et pour le psychologue de mieux connaître sa problématique...

A compléter avant l'expérimentation.

* *Fiche signalétique de l'enfant.*

Nom : PENILLAUD
Prénom : Angilique
Age : 6 ans
Sexe : F.

Handicap moteur :
maladie de Little. se déplace avec déambulateur.

Handicap sensoriel :

Non.

Handicap mental (aucun, léger, modéré, sévère) :
Niveau verbal et raisonnement moyens.
Trouble de la représentation de l'espace

Troubles associés :

Non

Troubles du langage :

non.

* *Fiche signalétique du moniteur.*

Nom : LATTY
Prénom : Laurence
Fonction : Stagiaire psych.

T du BARBAT Françoise psychologue

* *Objectifs poursuivis.*

A première vue, quels sont vos objectifs d'utilisation de ce logiciel avant sa mise en application ?

A. Du point de vue du savoir-faire (champ psycho-moteur) : motricité, perception,...

B. Du point de vue du savoir-être (champ socio-affectif) : affectivité, créativité, autonomie, communication, ...

C. Du point de vue ludique :

D. Autres : Evaluation du logiciel { technique et intérêt

Commentaires : Les deux choses qui ont dominé à la 1ère séance ont été 1) le triomphe de l'accès à l'ordinateur, jumelé avec celui de la rapidité à comprendre et utiliser le logiciel.

2) l'identification évidente avec la petite fille, ses "parents" et son "frère".

A la 2ème séance comme à la 1ère, la seule activité des personnages : chacun va s'asseoir quelquepart : elle exprime comme Marie Laure son impuissance de petite handicapée motrice.

Elle exprime beaucoup plus que celle-ci sa frustration, l'agressivité qui en résulte : elle "cogne" les personnages contre les "murs", fait sortir rageusement les personnages de la scène, en commentant "j'en veux plus du papa" par exemple, enfin "jette" les scènes qui ne lui sont pas agréables.

5ème séance = identification aux personnages identique. Lassitude : elle a déjà tout utilisé de tel. Pourtant elle est calme aujourd'hui. Cette réaction reflète plus semble-t-il de son impuissance et de sa frustration, que de son désintérêt pour le logiciel. Elle a fait une 5ème séance, et en demande d'autres.

* Remarque qui vaut aussi pour Marie Laure, Ray Tzu, et Denis : Ils savent à quoi sert la commande [suivant] mais ne comprennent pas vraiment la différence avec [précédent]

* *Variations des objectifs après utilisation.*

Quels sont vos objectifs de départ qui ont été satisfaits ?

Evaluation intéressante puisqu'il s'agit d'un des enfants les plus jeunes. Or. (1) elle parvient à maîtriser l'outil - Il y a eu apprentissage - en particulier sur le plan spatial.
(2) elle se projette énormément dans les scènes créées..

Percevez-vous de nouveaux objectifs? 3 utilisations possibles :

- Utilisation psychothérapeutique possible.

- Utilisation ludique (mais pas les 2 utilisations : il y a un choix à faire!)

- Sur le plan cognitif, on pourrait travailler la structuration de l'espace...

A compléter avant l'expérimentation.

* *Fiche signalétique de l'enfant.*

Nom : RODGE
Prénom : Blanche
Age : 6 ans
Sexe : F

Handicap moteur :
maladie de Little + orthèse
infectieuse. marche précoce.

Handicap sensoriel :
/

Handicap mental (aucun, léger, modéré, sévère) :
niveau verbal et raisonnement moyen
trouble de la représentation
de l'espace

Troubles associés :
Troubles de la personnalité.

Troubles du langage :
/

* *Fiche signalétique du moniteur.*

Nom : VEILLET et F. de BARBOT
Prénom : Patricia
Fonction : Psychologue

* *Objectifs poursuivis.*

A première vue, quels sont vos objectifs d'utilisation de ce logiciel avant sa mise en application ?

A. Du point de vue du savoir-faire (champ psycho-moteur) : motricité, perception,...

B. Du point de vue du savoir-être (champ socio-affectif) : affectivité, créativité, autonomie, communication, ...

C. Du point de vue ludique :

D. Autres : Evaluation des logiciels sur le plan de l'accessibilité et de l'intérêt pour cet enfant...

lentaires :

Séance n°2

La coordination visuo-motrice est difficile avec des sensations kinesthésiques non intégrées.

Pendant cette séance ses mains au lieu de l'écran pour cibler le curseur. Un jeu et la mesure du déplacement de la souris, elle intègre la représentation visuelle et kinesthésique du track-boule et peut se concentrer sur l'écran. Elle le manipule déplacement peut parvenir à cibler conjointement les cases correspondantes. Elle comprend qu'il est nécessaire de relâcher le bouton du bas afin de donner des ordres au logiciel.

A la fin de la séance, elle comprenait comment se servir de la souris, comment changer de ligne et comment faire bouger les remorquages.

L'orientation spatiale lui pose problème (↓ va à gauche, ← va descendre) mais une assimilation entre symbole et représentation visuelle pourrait s'élaborer. La ligne commande deviendrait plus accessible à sa compréhension. Elle est capable d'intégrer un enchaînement logique sans être perturbée par la multiplicité des opérations.

Séance n°2

Lucie choisit la petite fille et le petit garçon.
Elle fait avancer la P.F le long de l'écran - s'arrête à gauche de l'écran - Le P.G est assis, elle veut lui faire bouger les bras " il n'a pas le droit de se lever "... " La flèche il faut qu'elle s'en aille "
La scène s'appelle Franck

Autre séance : longue négociation pour choisir les personnages suis opté pour : l'écran, papa et le chien -
" qu'est-ce que je vais lui faire faire ? celui-là ! " (le papa)
(Papa immobile contre écran)
va chercher la maman et la fait entrer sur scène -
" Ils se donnent un bisou, je veux qu'ils marchent tous les 2 en même temps ... " finalement la fait avancer mon par un - Bientôt sa plaise sur le personnage de la maman " maman je vais te mettre par terre ... ici de punition ... " ou tu restes couché, elle ne peut rester couchée tout le temps " " Couchée par terre ! " " Je l'ai fait tomber ... "
" Je veux remuer les bras, papa remue les bras " ensuite veut le faire marcher
Fait apparaître le chien : " avec la maman, le chien ! "
(fait avancer le chien et la maman)
Lui veut les déplacer : déplace la maman vers le coin gauche de l'écran - Range le chien à droite de l'écran (ne dis pas rapport à la maman)
" Je veux les entendre tous les deux, faire des bisous ? "

Louise envoie le chien et le dirige vers la maman, dans
le chien au milieu de l'écran -
fait monter "au plafond" le chien, la maman aussi,
la papa toujours au milieu de la scène -
Vent arrêter,

Scène n°3

La scène débute avec le papa et la maman à l'écran - Aurélie fait disparaître la maman et garde le papa, qu'elle ne tarde pas à faire disparaître aussi.
Dans la scène suivante, elle choisit 3 loups et 1 chien - Elle fait subir différents mouvements aux loups mais n'établit pas de relations entre eux et avec le chien - Je lui demande ce que font les loups "ils vont écouter de la musique" - Elle avait un peu peur de devoir prendre des initiatives toute seule ("Je veux faire la même chose que Franck")
La dernière scène fait apparaître le père, la mère et 2 petites filles mais Aurélie ne garde que la mère ("Je veux la maman toute seule") à laquelle elle ne fait rien faire -

Au point de vue de l'utilisation de l'animateur, elle commence à très bien savoir s'orienter dans l'espace avec les touches de la ligne des bas, elle sait changer de ligne, et faire apparaître/disparaître les personnages - On observe également l'apprentissage d'une motricité plus fine -

Scène n°4

La première scène voit apparaître 4 petites filles - Aurélie en couche une par terre, en assoit une autre, fait couper les bras à la troisième -
Je lui demande ce qu'elles font ensemble - pas de réponse -
Elle aligne les 4 petites filles - En fait assoir une -

[redacted]

La fait monter au plafond - rit

" " couche à plat ventre - s'amuse beaucoup -

Amélie veut appeler la scène Franck ou Sylviane ou Jules -

Finalement l'appelle Sylviane -

Je lui demande si elle veut arrêter - Non. - J'enchaîne :

"Elles (les petites filles) vont se faire un bisou"

A : "J'arrête"

Elle éprouve quelques difficultés à se repérer dans l'espace du logiciel - (↑ confondue avec →)

Scène n°5

Amélie choisit le papa, la maman et le petit garçon - mais n'entre en scène que 3 personnages : le père, la mère et un petit garçon -

La maman est dirigée vers le coin gauche de l'écran, petit garçon vers le coin droit (ils sont dos à dos) - père est au milieu -

= "Papa va monter là-haut" (crayeux)

= Il est monté jusque là, il est descendu sur le (soleil) fait monter la maman en haut, vient s'a au milieu (en dessous du père) rapprochement père/mère Amène le petit garçon entre le père et la mère.

= Papa va filmer la maman et le petit garçon"

Veut appeler la scène Cécile - Franck - Sabine - Amélie

* *Variations des objectifs après utilisation.*

Quels sont vos objectifs de départ qui ont été satisfaits ?

- La maitrise des commandes du logiciel est précaire....
-

Percevez-vous de nouveaux objectifs?

- un travail sur le plan cognitif (en particulier structuration de l'espace) pourrait être entrepris.

A compléter avant l'expérimentation.

* *Fiche signalétique de l'enfant.*

Nom : LEKINA
Prénom : Fabure .
Age : 6 ans .
Sexe : M

Handicap moteur :

maladie de Little (marche avec cannes)

Handicap sensoriel :

/

Handicap mental (aucun, léger, modéré, sévère) :

Léger - - -

Troubles associés :

/

Troubles du langage :

/

* *Fiche signalétique du moniteur.*

Nom : VEILLET et F. de BARBOT
Prénom : Patricia
Fonction : Psychologue

* *Objectifs poursuivis.*

A première vue, quels sont vos objectifs d'utilisation de ce logiciel avant sa mise en application ?

A. Du point de vue du savoir-faire (champ psycho-moteur) : motricité, perception,...

B. Du point de vue du savoir-être (champ socio-affectif) : affectivité, créativité, autonomie, communication, ...

C. Du point de vue ludique :

D. Autres

Evaluation du logiciel sur le plan de l'accessibilité et de l'intérêt pour cet enfant...

* *Variations des objectifs après utilisation.*

Quels sont vos objectifs de départ qui ont été satisfaits ?

Le logiciel s'est révélé accessible à cet enfant – avec beaucoup de peine – et un travail d'apprentissage a été réalisé.

Percevez-vous de nouveaux objectifs?

Travail possible sur le plan cognitif (structuration de l'espace en particulier) ~~et~~ – et sur le plan de la créativité.

Commentaires : Fabrice a beaucoup de mal, à la 1ère séance, à intégrer les règles d'utilisation, même les plus simples. Beaucoup de mal, encore à la 2ème séance, à repérer les commandes. Il confond ↑ et ↓. A peu de faire se croiser 2 personnages "elle va pousser la petite fille".
Difficultés de spatialisation évidentes.
Mais aussi d'assimilation de données si simples (il faut cliquer à droite, et 2 fois).
3ème séance : ça va mieux. Mais les 2 garçons, le papa, et le grand père ne font rien "ils restent comme ça"
4ème séance. meilleure utilisation mais peu de résultats.
5ème séance. Il demande toujours à faire de l'architecture mais n'en fait pas grand-chose grand il est devant, à la fois par manque de moyens techniques et par manque d'imagination. Ce qui importe pour lui est d'avoir un adulte et un ordinateur pour lui tout seul.

A compléter avant l'expérimentation.

* *Fiche signalétique de l'enfant.*

Nom : BOURDAIS
Prénom : Romain
Age : 6 ans.
Sexe : M

Handicap moteur :
maladie de Little - Désambulation autonome...

Handicap sensoriel :
/

Handicap mental (aucun, léger, modéré, sévère) :
Niveau de raisonnement supérieur à la moyenne.
~~Fb~~ Trouble de la représentation de l'espace.

Troubles associés :
Agitation, et recherche des limites entraînant un comportement difficile.... Trouble de la personnalité....

Troubles du langage :
/

* *Fiche signalétique du moniteur.*

Nom : VEILLET et F. de BARBOT
Prénom : Fabrice
Fonction : Psychologue

* *Objectifs poursuivis.*

A première vue, quels sont vos objectifs d'utilisation de ce logiciel avant sa mise en application ?

A. Du point de vue du savoir-faire (champ psycho-moteur) : motricité, perception,...

B. Du point de vue du savoir-être (champ socio-affectif) : affectivité, créativité, autonomie, communication, ...

C. Du point de vue ludique :

D. Autres

Evaluation du logiciel sur le plan de l'accessibilité et de l'intérêt pour cet enfant...

Commentaires :

Séance n° 1

Au premier abord, on peut observer une coordination visuo-motrice difficile en ce qui concerne le maniement de la souris. Franck a du mal à cibler avec la souris. Cette intégration des informations tactiles et kinesthésique à la vue ne sera pas réalisé à la fin de la séance.
Il ne comprend pas le sens des flèches (← ; →) et il lui est très difficile de comprendre un enchaînement logique de 2 opérations à effectuer (ex : pour faire apparaître 1 personnage : amener la souris tout à fait à gauche pour faire apparaître la tête du personnage. 2e opération : amener la souris sur la 3e case à partir de la gauche pour que le personnage puisse venir en totalité sur la scène)
Lors de cette première séance, il n'est possible de travailler qu'avec 1 seule ligne de commande.

Franck paraît utiliser l'appareil de façon un peu magique : il lui crie les ordres et attends que les opérations se fassent seules. Il lui faut remarquer que l'ordinateur exécute les ordres qu'on lui donne par la ligne du bas et grâce à l'intermédiaire de la souris. L'impulsivité caractérise cette séance. Il manipule brutalement la souris et veut passer immédiatement à l'autre personnage sans avoir compris la manipulation de la.

Séance n°2

A cette séance, Franck paraît moins encadré par les inj
tions et malgré l'agitation de la 1ère séance, un apprentissag
s'est effectué :

- Il comprends le sens des flèches qui permettent de fai
bouger les personnages ;
- Il sait faire apparaître / disparaître le personnage de l'
cran ;
- Il expérimente la souris sur les personnages.
- " comprends comment changer de ligne.

Je lui montre comment sélectionner ses personnages ce qui
semble très difficile pour lui étant donné la multiplicité
commandes et le fait qu'il ne sache pas lire. Il
peut se repérer qu'en mémorisant la symbolisme de
sins de la ligne du bas.
Il prend plaisir à associer ses personnages et à assimiler
méthodologie pour effectuer cette opération.

Du point de vue clinique, le effectue des groupes avec le
sonnages : papa d'un côté avec le chien, maman de l'
aussi avec son chien et les personnages des vidéos. Que
t-il ? ils sont punis, ils ont touchés aux ordinateurs -
(c'est grave !), ils peuvent toucher les prises et ils ne
L'idée de changement d'un interdit apparaît dans
scène.

Séance n°3

Les acquis des séances précédentes sont toujours là - Cependant Franck est perdu pour sélectionner ses personnages : il confond prendre avec o.k - Il ne paraît pas comprendre ce que signifie "enregistrer" : l'idée et l'image restent abstraites pour lui même si on lui montre des scènes enregistrées précédemment -

Du point de vue clinique, c'est 2 groupes que l'on observe à nouveau : papa, maman, petite fille tassés ensemble d'un côté et le bras de l'autre - (" Ils sont tous fatigués ") -
Le bras va à table, remue les bras au-dessus du pain et de la mie aussi pour dire (" Ils sont punis parce qu'ils ont pris l'ordinateur ; ils sont sortis de là. - l'écran - pour prendre l'ordinateur ")
Les personnages se retrouvent en haut de l'écran (" pour pas qu'ils touchent à l'ordinateur ") Enfin le papa et la petite fille redescendent pour le faire un bisou.
Nom de la scène : a a aeee (tapé au hasard sur le clavier)

Séance n°4

Franck ne comprends pas le sens des flèches ← ; ⇒ qui permet de choisir décors ou personnages ; les notions de suivant/précédent ne sont pas acquises - En ce qui concerne l'écran de choix des personnages, il ne comprends pas que la fenêtre de gauche ou apparaissent les personnages, permet d'effectuer un choix ; pour

lui, le fait qu'ils (les personnages) soient là signifie qu'
ont été sélectionnés et amène des questions comme : "pou
il y a 2 loups (le loup qui est présenté par la fenêtre et
qui est sélectionné)_ Franck va effacer ses personnages pa
qui va avoir l'impression qu'ils sont en double_

La première scène se passe dans la forêt et met en scène
loup, le chien, la petite fille et le papa
le papa est couché, le loup aussi ("mange le papa")
petite fille aussi_
= Ils (personnages) se fabriquent, ils ne veulent jamais
jouer avec eux"
(Il fait une remarque sur ma voix qu'il trouve chan
tillonnant)
(N.B. Ici il a été amené à poser des interdits, car Fran
me touchait de façon répétée)
2ème scène : "Je n'aime pas la forêt" (choisit la salle -
je ne veux que ça), je ne veux pas de personnages
(joue avec la souris, la positionne sur une chaise.
"La souris est assise ; elle ne veut pas travailler"
Franck ne se souvient plus des personnages qu'il
("j'avais combien de loups?")
"Je veux l'emmener jusque là (le papa) mais si je
me bat, il va être puni (?) parce qu'il ne veut pas
(?) le papa, "il dort" (papa dehors) il dort en ayant
contre le mur (?) c'est ma nouvelle peur"
Peur : n'aime pas le dessin - rit
Peur : signe le loup, la petite fille et le chien les
uns les autres
= Le " mange les pieds (du papa), les rideaux

Séance n°5

1ère scène : Franck choisit la forêt, la maman, le papa, un chien.
Lors de la création de la scène, il ne fait apparaître que la mère "Je ne veux qu'il y ait personne"
Il dit à propos de la mère "Elle se déplace", le fait avancer à droite, puis à gauche = Comment se fait qu'elle est dans le mur ? voilà qu'elle est punie là-haut "En arrière !"
S'arrête et veut absolument donner un nom à la scène "Franck" ou non le tien, je pourrais l'appeler Sylviane"
Dès le début de la scène, pense au nom qu'il va donner à la scène.

2ème scène : Petite fille - loup - 2 chiens - Salle-à-manger
"Petite fille à peur des chiens et des loups... elle reste seule ... Lui en a l'accord !"
"Elle ne sait pas la porte (porte à droite de l'écran)... Je veux qu'elle sorte par la porte.... Je veux qu'elle aille derrière la porte"
Obligé la scène pour pouvoir donner immédiatement un nom à la scène : Sylviane.

Du point de vue de l'utilisation du logiciel, Franck comprend qu'il va falloir sélectionner un décor, des personnages, puis créer un scénario. Il est désemparé par le fait de ne pas savoir lire la signification des dessins pour un rôle dominant.

En ce qui concerne l'écran de choix de personnage, je ne comprends pas que la case de droite sert à choisir le personnage – Si par exemple, il choisit de coup, il ne va pas comprendre pourquoi il y a 2 coups (le modèle + celui qu'il a pris) il va effacer celui qu'il a choisi afin qu'il n'y ait qu'un seul coup présent sur l'écran –

J'essaie de lui expliquer que si je vois Franck et la flèche Franck ce n'est pas la même chose – Il n'y a qu'un seul Franck et non pas deux –

Les notions de suivant (→) et précédant (←) au regard ceyx, ne sont pas assimilées –

Du point de vue numérique, il comprends parfaitement qu'il a 6 personnages à choisir, lorsqu'il en a choisi un qu'il en reste 5 etc....

* *Variations des objectifs après utilisation.*

Quels sont vos objectifs de départ qui ont été satisfaits ?

Frank a eu bcp de mal à maîtriser les commandes du logiciel - De plus l'aspect projectif du matériel semble angoissant pour lui, avec perte de distance ....

Percevez-vous de nouveaux objectifs?

Contre indication.

A compléter avant l'expérimentation.

* *Fiche signalétique de l'enfant.*

Nom : MIHOUBI
Prénom : Karim
Age : 9 ans.
Sexe : M

Handicap moteur : Hémiplégie droite.

Handicap sensoriel :

/

Handicap mental (aucun, léger, modéré, sévère) :
Raisonnement moyen.

Troubles associés :
TDI de l'attention et du comportement.

Troubles du langage :

/

* *Fiche signalétique du moniteur.*

Nom : LATTY + F. de BARBOT

Prénom : Laurence

Fonction : Stagiaire psychologue - psychologue

* *Objectifs poursuivis.*

A première vue, quels sont vos objectifs d'utilisation de ce logiciel avant sa mise en application ?

A. Du point de vue du savoir-faire (champ psycho-moteur) : motricité, perception,...

B. Du point de vue du savoir-être (champ socio-affectif) : affectivité, créativité, autonomie, communication, ...

C. Du point de vue ludique :

D. Autres : Evaluation du logiciel

Commentaires : Karim, très discret de "jouer à l'ordinateur" – il me l'a demandé 20 fois – a du mal à porter son attention de façon suivie, s'impatiente avant de chercher à comprendre. Son attention est distraite facilement, par n'importe quoi. Quand il a eu le souris mieux du logiciel, il n'a pas eu en fait grand chose : à part les personnages à traîner entre eux, souvent même pas tout, il avait la souris. Cependant le fauteuil roulant a été pour lui source de grand plaisir, il en actionnait plusieurs en même temps, et expliquait que c'était son ami Edouard. Puis il leur a donné des prénoms toujours différents, apparemment des enfants qui ne sont pas du service. Il a beaucoup utilisé le mode de déplacement des personnages avec la souris, mais de façon erratique : le personnage court partout sans but ni activité organisée.

* *Variations des objectifs après utilisation.*

Quels sont vos objectifs de départ qui ont été satisfaits ?

Maitrise de l'outil : +.
Utilisation avec discontinue, en raison des TBL de l'attention et du comportement.

Percevez-vous de nouveaux objectifs?

Je n'envisage pas d'autre utilisation avec Koïm pour le moment

A compléter avant l'expérimentation.

* *Fiche signalétique de l'enfant.*

Nom : FELIX
Prénom : Denis
Age : 9 ans
Sexe : M

Handicap moteur :

aucun.

Handicap sensoriel :

aucun

Handicap mental (aucun, léger, modéré, sévère) :

léger.

Troubles associés :

non

Troubles du langage : ~~Aphasie congénitale~~
Dysphasie...

* *Fiche signalétique du moniteur.*

Nom : LATTY + de BARBOT
Prénom : Laurence Françoise
Fonction : stagiaire Ψ psychologue

* *Objectifs poursuivis.*

A première vue, quels sont vos objectifs d'utilisation de ce logiciel avant sa mise en application ?

A. Du point de vue du savoir-faire (champ psycho-moteur) : motricité, perception,...

B. Du point de vue du savoir-être (champ socio-affectif) : affectivité, créativité, autonomie, communication, ...

C. Du point de vue ludique :

D. Autres : Dans cette étape :
Evaluation du logiciel { technique et intérêt.
Observation de l'enfant.

Commentaires : 1ère séance = l'identif' au garçon est manifeste
Ensuite il y a eu identification au petit chien.

Denis a su très rapidement se servir du logiciel, avec beaucoup d'efficacité.

Les séances ont été très espacées pour lui, parceque la demande originelle qu'il en avait, très vive, s'est trouvé rapidement contrecarrée par une appréhension montante face à ces séances.
Entre la 1ère et la 2ème : demandes réitérées, et comme par hasard je ne le trouvais pas disponible quand je l'étais. Entre la 2ème et la 3ème il y a eu fuite masquée par une attitude très ludique. Puis fuite plus nette entre la 3ème et la 4ème séance, et même refus, malgré le fait que ça l'ennuyait manifestement d'avoir à me dire "non" — car Denis est un enfant très coopérant, et qui veut donner une bonne image de lui-même.
A la 4ème séance, que j'ai exigée, en lui promettant que ce serait la dernière, il n'a pas créé de scène, il a seulement regardé. Il avait peur de pouvoir dire de quoi :
A mon interprétation qu'il a peur de ce qu'il voit sur l'écran, car c'est lui dans la vie, en dehors de ce service hospitalier et de sa famille, il a dit non, n'était pas cela. On en a beaucoup [illegible]

* *Variations des objectifs après utilisation.*

Quels sont vos objectifs de départ qui ont été satisfaits ?

Evaluation des logiciels : accessibilité +
Observation d'un enfant
qui ne parle pas : le matériel qui
a surgi a permis une interprétation,
ce qui dépassait les objectifs de
départ.

Percevez-vous de nouveaux objectifs?

Denis fait partie des enfants
pour lesquels une utilisation
psychothérapeutique est envisageable

A compléter avant l'expérimentation.

* *Fiche signalétique de l'enfant.*

Nom :
Prénom : Virginie
Age : 11 ans
Sexe : F

Handicap moteur :

aucun.

Handicap sensoriel :

aucun.

Handicap mental (aucun, léger, modéré, sévère) :

Handicap modéré

Troubles associés :

/ non.

Troubles du langage :

trouble de l'articulation bucco faciale rendant le langage presque incompréhensible.

* *Fiche signalétique du moniteur.*

et F. de BARBOT

Nom : MEILLOT
Prénom : Patricia
Fonction : Psychologue

* *Objectifs poursuivis.*

A première vue, quels sont vos objectifs d'utilisation de ce logiciel avant sa mise en application ?

A. Du point de vue du savoir-faire (champ psycho-moteur) : motricité, perception,...

B. Du point de vue du savoir-être (champ socio-affectif) : affectivité, créativité, autonomie, communication, ...

C. Du point de vue ludique :

D. Autres : Evaluation du logiciel :
- accessibilité
- interet pour un enfant ayant tres peu de communication orale.

Séance n°1

Au cours de cette séance, on peut observer que le plaisir de Virginie est essentiellement un plaisir cognitif. Elle retire beaucoup de satisfaction à apprendre à maîtriser le logiciel. Elle accepte d'être guidée verbalement mais veut rester aux "commandes" de l'appareil.
La compréhension du fonctionnement du logiciel ne paraît pas lui poser de problèmes.
Elle n'utilise pas la souris pour faire bouger les personnages, et jette systématiquement les scènes créées.

Seance n°3

Lors de cette séance, un certain laisser-aller s'installe : La précision recherchée auparavant (pour faire associer les personnages par exemple) disparaît comme si le fait d'être concerné à une bonne compréhension du logiciel le rendait ininteressant et dispensait désormais de tout effort. L'aspect cognitif des premières séances cède le pas à un aspect beaucoup plus ludique : Virginie rit beaucoup et s'amuse énormément à voir les personnages se déplacer rapidement, voler ou tourner autour d'eux comme des girouettes. Cependant, ce côté ludique s'épuise beaucoup plus vite que la recherche de la maîtrise cognitive de l'appareil.

du point de vue du fonctionnement de l'appareil :

- Si on ne lui rappelle pas qu'il y a une commande enregistre elle ne s'en sert pas ;
- Virginie se répète un peu plus d'après le texte que d'après les dessins ;
- Elle n'utilise pratiquement pas la souris pouvant faire bouger les personnages de façon beaucoup plus souple - ;
- Virginia aime donner un nom à la scène ;
- Elle n'utilise pas les commandes "modifier" et "Renomm
- La musique lui manque beaucoup.

Enfin elle continue de jeter un œil et veut jeter celle des cach

* *Variations des objectifs après utilisation.*

Quels sont vos objectifs de départ qui ont été satisfaits ?

- Maîtrise rapide du fonctionnement du logiciel
- Virginie s'en sert peu pour exprimer quelque chose. Investissement ludique, rapidement épuisé..

Percevez-vous de nouveaux objectifs?

Je ne pense pas réutiliser Orchithèque avec Virginie pour le moment.

A compléter avant l'expérimentation.

* *Fiche signalétique de l'enfant.*

Nom :
Prénom : Frédéric
Age : 11 ans
Sexe : M.

Handicap moteur :

/

Handicap sensoriel :

/

Handicap mental (aucun, léger, modéré, sévère) :

Léger.

Troubles associés :

/

Troubles du langage :

Syndrome pseudo bulbaire ; TCL
parole très déformée + difficulté d'évocation des mots.

* *Fiche signalétique du moniteur.*

Nom : VEILLET et F. de BARBOT
Prénom : Patricia
Fonction : Psychologue

* *Objectifs poursuivis.*

A première vue, quels sont vos objectifs d'utilisation de ce logiciel avant sa mise en application ?

A. Du point de vue du savoir-faire (champ psycho-moteur) : motricité, perception,...

B. Du point de vue du savoir-être (champ socio-affectif) : affectivité, créativité, autonomie, communication, ...

C. Du point de vue ludique :

D. Autres : Evaluation du logiciel sur le plan de l'accessibilité et de l'intérêt pour cet enfant....

Commentaires :

Séance n° 1 : D'une façon générale, l'enfant prend beaucoup de plaisir à apprendre à maîtriser cet outil ce qu'il réussit en un laps de temps assez court – En outre, on peut constater durant la séance les éléments suivants :

1) Du point de vue du fonctionnement du logiciel :

- il cherchait absolument la musique
Le nombre de décors lui paraissait peu élevé et il cherchait celui qui symbolisait cette notion.
- Il voulait faire bouger les éléments statiques (par ex. ouvrir la porte de l'un des décors)
La souris était beaucoup sollicitée s/les personnages
Il comprenait le déroulement logique des opérations à effectuer (« plaisir à comprendre »)
Pour faire défiler les décors, il n'utilisait que ← ou → mais pas alternativement l'un et l'autre –

2) Du point de vue cognitif :

L'enfant cherche à faire effectuer aux personnages des actions complexes (les faire asseoir) et montre dans l'utilisation du logiciel un certain esprit pratique (par ex. il aime que le loup se couche au plafond, pour éviter plusieurs manips afin qu'il se retrouve au bon endroit, il le fait sortir de là et ne plus rentrer).
Il ne comprenait pas pourquoi les personnages arrivés à la lisière de l'écran ne pouvaient plus être avancés
Le déplacement des personnages sur les associé, obligeait

l'enfant à comprendre le sens de flèche figurant au bas d
l'écran et donc à s'orienter spatialement en établissa
une connexion entre personnages et symboles, puis à adopte
une stratégie.

5) <u>du point de vue clinique</u>

On note beaucoup d'agressivité (Loup qui se cogne la tes
contre la bordure de l'écran, petit garçon assis sur la
du tronc de l'arbre, petite fille allongée qui se cogne
tête contre le tronc de l'arbre) - Peut-être un enja
maîtrise de l'anxiété (à confirmer dans les séances
venir) : l'enfant fait apparaître le loup - pousse des
le fait disparaître et cela plusieurs fois de suite - Il
éprouve également un besoin de maîtrise très grand,
le plan cognitif -

Noms des zones : Hypo - Loup - Jupe - Loup.

Séance n°2

A cette séance, Frédéric a montré moins d'intérêt tout en complétant ses connaissances du logiciel qui n'étaient pas parfaites comme l'a démontré les 35' passées ensemble

La fonctionnement du logiciel

Frédéric prends plaisir à explorer les modes qu'il ne connaît pas bien comme par exemple "Modifier" le personnage. Il va de lui-même cliquer sur modifier, supprimer un à un les personnages choisis puis les réintroduire, satisfait d'avoir compris de lui-même un aspect du logiciel. Cependant, il a besoin d'aide pour 2 séquences non comprises :

- Lorsque l'ordinateur demande une confirmation d'ordre (OK ou annulation) il ne comprenait pas la signification "d'annulation"
- pour ces mêmes confirmation d'ordre, il ne cliquait pas sur la ligne du bas, mais en haut sur l'ordre donné précédemment.

En outre, lors des opérations de sélection (décors, personnage, musique, voix, stop) il confond souvent voix avec stop. Il oublie l'enregistrement bien qu'il prenne plaisir à revoir les scènes créées et regrette encore cette fois-ci qu'il n'y ait pas de musique.

N.B : Lors de la sélection de départ (décors-personnages, musique-, crédits) il me comprend pas la symbolisme des dessins et se demande où est le dessin qui représente le décor, où est le petit garçon qui représente la marionnette.

Noms des scènes : — Jaman, papa, petit garçon, loup
— Jaman 12
— Papa

Scène n°3

Lors de cette brève scène, un côté ludique émerge (faire asseoir le loup sur la table) et si Frédéric est beaucoup moins intéressé il n'en apporte pas moins un certain soin à l'élaboration de la scène.

Variations des objectifs après utilisation.

Quels sont vos objectifs de départ qui ont été satisfaits ?

- maîtrise rapide du fonctionnement des logiciels
- cet aspect de maîtrise et de découverte a motivé Frédéric.
- il s'est projeté dans les scènes créées...

Percevez-vous de nouveaux objectifs?

pas pour le moment...

A compléter après l'ensemble des séances d'expérimentation.

* *Fiche signalétique du moniteur.*

Nom : de BARBOT
Prénom : Françoise
Fonction : Psychologue

Nombre d'enfants vus : 15 (11 IMC + 4 dysphasiques) de 5 à 12 ans.

Nombre de séances par enfant : de 3 à 5

N.B. Evaluation portant sur une version non terminée (absence de musique .... nombre de personnages incomplet).

* *Critique du logiciel.*

A. Sur le plan technique :

- possibilité de réglage des paramètres :

Grand intérêt de la planification des tableaux de commandes.
Intérêt de pouvoir enregistrer les paramètres sélectionnés pour un enfant donné

- facilité d'utilisation (mise en route, menus, choix, adaptation à divers handicaps, possibilité d'annuler en cas d'erreur,...) :

Accessible à tous les degrés de handicap moteur.
Utilisation facile.
Un problème majeur : lorsque l'enfant a sélectionné plusieurs fois le même personnage, il est impossible de savoir lequel de ces personnages on est en train de commander : s'il y a 3 loups ou 4 papas, on est complètement perdu. Ce qui s'est présenté à plusieurs reprises

- qualité du graphisme (notamment la dimension des caractères et des dessins), de l'animation, {du son}; problèmes liés à la taille de l'écran, au noir et blanc, ... :

- Graphisme excellent.
- Nous n'avons pas rencontré de problèmes liés à la taille de l'écran ou à l'absence de couleur.
- Un détail: lorsque l'on utilise le déplacement des personnages par flèches, on ne parvient pas à les asseoir avec précision sur les chaises, etc...
- Plus gênant: les icônes de la 3e ligne des commandes sont d'une taille différente, ce qui entraîne une place différente et des erreurs de manipulation.

- autres remarques :

- Commandes difficilement comprises par notre population :- ← précédent (par opposition à → suivant)
- ↩ ↪ : si l'enfant comprend bien qu'il s'agit de tourner le personnage, le sens est souvent mal compris
- Le fait que l'enfant puisse, au cours d'une scène, arrêter l'enregistrement nous paraît un inconvénient : il y a en sauvegarde des erreurs de manipulation, non corrigées et entraînant la perte de la scène.

B. Sur le plan pédagogique et thérapeutique :

- utilité, comparaison avec le(s) système(s) utilisé(s) jusqu'alors :

Utilisations qui nous ont paru intéressantes
- travail sur la structuration de l'espace
- utilisation ludique, en particulier pour les enfants à handicap extrême : possibilité d'être actif et autonome dans le jeu - utilisation d'un espace transitionnel - possibilité de maîtrise des effets à travers le jeu
- utilisation projective = l'outil est particulièrement puissant dans ce domaine, il a permis pour les enfants l'expression des problématiques que nous n'avions pas perçues dans les situations habituelles et même dans des situations favorables à cette mise à jour (dessins, jeux etc...) Il s'agit donc d'un outil ayant des possibilités diagnostiques et psychothérapeutiques très réelles. Dans les situations psychothérapeutiques le rôle des verbalisations, des associations et du transfert demeure essentiel.

La puissance de l'outil constitue une contre indication dans certains cas (troubles de la personnalité par exemple)

- adaptabilité à différents types d'utilisateurs :
- utilisable par toutes les enfants que nous avons vus (y compris ceux à handicap extrême)
- utilisation nécessitant un apprentissage important pour les 5-7 ans ayant des troubles praxiques.
- à partir de 9-10 ans, l'intérêt s'épuise vite une fois que l'enfant a maîtrisé toutes les possibilités du programme (~~maîtrise~~ exploration et maîtrise qui le motive beaucoup) les possibilités créatives sont limitées pour cette classe d'âge - et peut-être les personnages proposés trop enfantins
- par contre pour les 5-9 ans les utilisations sont multiples - Le Logiciel est particulièrement adapté à cette tranche d'âge....

* *Propositions d'améliorations :*

- Nécessité d'identifier le personnage que l'on est en train de commander lorsque plusieurs exemplaires du même personnage ont été selectionné.
- Y aurait-il moyen de faire jouer deux enfants en même temps en double commande ?

ANNEXE 4 :

Code source du programme.

```
unit messages_erreurs;

interface


 function erreur_ouverture_fichier (nom_fich : string) : boolean;
{ Pré : nom_fich contient le nom d'un fichier qui n'a pas pû être ouvert. }
{ Post : une fenêtre d'alerte a signalé à l'utilisateur que le fichier nom_fich est introuvable }
{        et lui a demandé s'il désirait chercher le fichier. }
{        Si l'utilisateur a décidé de chercher le fichier, la fonction retourne TRUE; }
{        s'il a décidé de continuer sans le chercher elle retourne FALSE. }

 procedure erreur_attitude_manquante (num_pers, num_attit : integer);
{ Pré : num_pers >=1 ; 1 <= num-attit <= 19 }
{ Post : une fenêtre a été affichée pour indiquer que l'attitude num_attit du personnage num_pers est introuvable }

 procedure erreur_systeme_musique;
{ Pré : / }
{ Post : une fenêtre a été affichée pour indiquer à l'utilisateur que le système utilisé ne permet pas de jouer la
       musique. }

 procedure erreur_memoire_accessoire;
{ Pré : / }
{ Post : une fenêtre a été affichée pour indiquer à l'utilisateur que la mémoire est insuffisante }
{        pour ouvrir l'accessoire de bureau demandé. }

 procedure erreur_ecriture_fichier (nom_fich : string;
           cause : OSErr);
{ Pré : nom_fich contient un nom de fichier dans lequel on n'a pas pû écrire }
{       et cause contient un numéro d'erreur du File Manager. }
{ Post : l'erreur a été signalée à l'utilisateur. }

implementation

 const
  err_ouv_fich_id = 150;
  err_attit_manq_id = 160;
  err_sys_mus_id = 170;
  err_mem_acc_id = 180;
  err_ecr_fich_id = 190;

 function erreur_ouverture_fichier;

  var
   continuer : integer;

 begin
  ParamText(nom_fich, '', '', '');
  continuer := StopAlert(err_ouv_fich_id, nil);
  erreur_ouverture_fichier := (continuer = 1);
 end; {erreur_ouverture_fichier}


 procedure erreur_attitude_manquante;

  var
   continuer : integer;
   pers_string, attit_string : str255;
```

```
 begin
  NumToString(num_pers, pers_string);
  NumToString(num_attit, attit_string);
  ParamText(attit_string, pers_string, '', '');
  continuer := StopAlert(err_attit_manq_id, nil);
 end; {erreur_attitude_manquante}

 procedure erreur_systeme_musique;

  var
   continuer: integer;

 begin
  continuer := StopAlert(err_sys_mus_id, nil);
 end; {erreur_systeme_musique}

 procedure erreur_memoire_accessoire;

  var
   continuer: integer;

 begin
  continuer := StopAlert(err_mem_acc_id, nil);
 end; {erreur_memoire_accessoire}

 procedure erreur_ecriture_fichier;

  var
   continuer: integer;
   cause_str: str255;

 begin
  case cause of
   DirFulErr:
    cause_str := 'Le dossier est plein';
   DskFulErr:
    cause_str := 'Le disque est plein';
   WPrErr:
    cause_str := 'Le disque est protégé';
   otherwise
    NumToString(integer(cause), cause_str);
  end;
  ParamText(nom_fich, cause_str, '', '');
  continuer := StopAlert(err_ecr_fich_id, nil);
 end;


end. {unit messages_erreur}
```

```
unit fichiers;

interface

 uses
  messages_erreurs;

 const
  max_lignes = 4; { nombre maximum de lignes de commandes d'animation pouvant être spécifiées dans les
       paramètres }
  max_com = 8; { nombre maximum de commandes dans une ligne de commandes }
  nbre_com_bloc = 256; { les scènes enregistrées sont découpées en blocs contenant nbre_com_bloc commandes
       }
  nbre_car_ligne_tr = 64; { nombre de caractères dans une ligne de trace }
  nbre_lignes_bloc_tr = 128; { les traces sont découpées en blocs contenant nbre_lignes_bloc_tr lignes }

{ les constantes suivantes sont utilisées pour désigner les différentes icônes du programme }
{ lorsque le nom est suivi de "t" : l'icône est composée d'un texte }
{                               "i" : l'icône est composée d'une image }
{                               "ti" : l'icône est composéee d'une image et d'un texte }
  regarder_scene_t = 1;
  creer_scene_t = 2;
  gauche_i = 3;
  droite_i = 4;
  ok_ti = 5;
  ok_t = 35;
  jeter_ti = 6;
  jeter_i = 36;
  jeter_t = 37;
  stop_i = 7;
  stop_t = 38;
  demande_stop_t = 45;
  annuler_ti = 8;
  recreer_ti = 9;
  recreer_i = 39;
  recreer_t = 40;
  musique_ti = 10;
  musique_i = 41;
  musique_t = 42;
  pause_ti = 11;
  decors_t = 12;
  personnages_t = 13;
  ecouter_ti = 14;
  change_pers_i = 15;
  change_com_i = 16;
  entree_sortie_i = 17;
  haut_i = 18;
  bas_i = 19;
  debout_i = 20;
  assis_i = 21;
  couche_i = 22;
  pivote_gauche_i = 23;
  pivote_droite_i = 24;
  agite_i = 25;
  enregistre_i = 26;
  precedent_ti = 27;
  suivant_ti = 28;
  pers_pas_select_i = 29;
  decor_pas_select_i = 30;
```

```
    musique_pas_select_i = 31;
    demande_nom_t = 32;
    arreter_regarder_t = 33;
    quitter_animation_t = 34;
    modifier_ti = 43;
    effacer_ti = 44;
    donne_nom_scene_t = 46;
    prendre_ti = 47;
    creer_t = 48;
    lapin1_i = 100;
    lapin2_i = 101;

  type
    str20 = string[20];
    str32 = string[32];
    Tcommandes = array[1..max_lignes, 1..max_com] of integer;
{ tableau contenant les lignes de commandes spécifiées dans les paramètres }
    Tparam_enfant = record { liste des paramètres liés à un enfant}
      nom_enf: str255; { nom de l'enfant }
      mode_util: integer; { mode d'utilisation : 1 = souris, 2 = clavier-flèches, 3 = clavier-défilement }
      vitesse_def: integer; { vitesse de défilement pour le mode clavier-défilement }
      commandes: Tcommandes; { lignes de commandes pour l'animation }
      type_trace: integer; { type de trace }
      enreg_autom: boolean; { TRUE si l'enregistrement automatique doit être branché }
    end;
    Tscene_ref = record { référence d'une scène enregistrée reprise dans le catalogue des scènes }
      nom_fich: str32; { nom du fichier contenant la scène }
      nom_scene: str32; { nom de la scène }
      num_decor: integer; {identifiant du décor }
      num_pers_1: integer; { identifiant du premier personnage }
      num_pers_2: integer; { identifiant du deuxième personnage }
      num_pers_3: integer; { identifiant du troisième personnage }
      num_pers_4: integer; { identifiant du quatrième personnage }
      num_musique: integer; { identifiant de la musique }
    end;
    Tcommande = record { représentation d'une commande }
      num_commande: integer; { identifiant de la  commande }
      num_personnage: SignedByte; { numéro (1..4) du personnage }
      attitude_pers: SignedByte; { numéro (1..18) de l'attitude }
      ligne_pers: integer; { coordonnée Y }
      colonne_pers: integer; { coordonnée X }
    end;
    Tbloc_commandes = array[1..nbre_com_bloc] of Tcommande; { blocs de commandes d'une scène enregistrée }
    Tligne_trace = packed array[1..nbre_car_ligne_tr] of char; { ligne de trace }
    Tbloc_trace = array[1..nbre_lignes_bloc_tr] of Tligne_trace; { bloc de trace }

  procedure ouvrir_fichiers;
{ Pré : / }
{ Post : les fichiers nécessaires à l'application sont ouverts }

  procedure fermer_fichiers;
{ Pré : / }
{ Post : les fichiers nécessaires à l'application sont fermés }

  procedure ecrire_parametres_enfant (parametres: Tparam_enfant);
{ Pré : parametres contient une liste de paramètres }
{ Post : si parametres.nom_enf = '', alors les parametres par defaut sont remplacés par parametres }
{        si parametres.nom_enf est connu, alors les parametres correspondant sont remplacés par parametres }
{        si parametres.nom_enf est inconnu, alors le record parametre est ajouté dans le fichier de parametres }
```

```
 function lire_parametres_enfant (nom_enfant : str255) : Tparam_enfant;
{ Pré : / }
{ Post : si nom_enfant = '' ou est inconnu, la fonction renvoie les paramètres par défaut }
{        sinon elle renvoie les parametres dont le champ nom_enf = nom_enfant }

 function lire_illustration_nr (numero : integer) : pichandle;
{ Pré : / }
{ Post : si numero identifie une icône du programme (voir liste des constantes), la fonction }
{        renvoie un handle vers cette icône, sinon elle renvoie nil }

 function lire_nbre_decors : integer;
{ Pré : / }
{ Post : renvoie le nombre de décors disponibles }

 function lire_decor_nr (numero : integer) : pichandle;
{ Pré : 1 <= numero <= lire_nbre_decors }
{ Post : la fonction renvoie un handle vers le decors spécifié par numéro }

 function lire_nbre_personnages : integer;
{ Pré : / }
{ Post : renvoie le nombre de personnages disponibles }

 function lire_personnage_nr_att (numero, attitude : integer) : pichandle;
{ Pré : 1 <= numéro <= lire_nbre_personnages }
{       1 <= attitude <= 19 }
{ Post : renvoie un handle vers l'attitude spécifiée par attitude du personnage identifié par numero }

 function lire_ecran_fixe_nr (numero : integer) : pichandle;
{ Pré : 1 <= numero <= 3 }
{ Post : renvoie un handle vers l'écran fixe spécifié par numéro }

 function lire_nbre_musiques : integer;
{ Pré : / }
{ Post : renvoie le nombre de musiques disponibles }

 function lire_ecran_musique_nr (numero : integer) : pichandle;
{ Pré : 1 <= numéro <= lire_nbre_musiques }
{ Post : renvoie un handle vers l'écran de présentation de la musique spécifiée par numéro }

 function lire_musique_nr (numero : integer) : handle;
{ Pré : 1 <= numéro <= lire_nbre_musiques }
{ Post : renvoie un handle vers la musique spécifiée par numéro }

 function lire_nbre_scenes_enf (nom_enfant : str255) : integer;
{ Pré : / }
{ Post : renvoie le nombre de scènes enregistrées par l'utilisateur nom_enfant }

 function lire_reference_scene_enf_nr (nom_enfant : str255;
          num_ref : integer) : Tscene_ref;
{ Pré : 1 <= num_ref <= lire_nbre_scenes_enf (nom_enfant) }
{ Post : renvoie la référence de scène numéro num_ref enregistrée par l'utilisateur nom_enfant}

 procedure ecrire_reference_scene_enf_nr (nom_enfant : str255;
          num_ref : integer;
          var reference : Tscene_ref);
{ Pré : 1 <= num_ref <= lire_nbre_scenes_enf (nom_enfant) + 1 }
{ Post : si num_ref <= lire_nbre_scenes_enf (nom_enfant) la référence de scène numéro num_ref }
{        enregistrée par l'utilisateur nom_enfant est remplacée par reference, }
```

```
{       sinon reference est ajoutée au catalogue des scènes enregistrées par l'utilisateur nom_enfant }
{       et reference.nom_fich contient un nouveau nom de fichier }

 procedure effacer_scene_enf_nr (nom_enfant: str255;
           num_ref: integer);
{ Pré : 1 <= num_ref <= lire_nbre_scenes (nom_enfant) }
{ Post : la scène enregistrée par l'utilisateur nom_enfant }
{       et dont la référence est identifiée par num_ref est supprimée }

 function lire_nbre_blocs_scene (nom_fich: str32): integer;
{ Pré : / }
{ Post : renvoie le nombre de blocs contenus dans la scène enregistrée}
{       dont le nom de fichier est nom_fich }

 function lire_bloc_scene_nr (nom_fich: str32;
           num_bloc: integer): Tbloc_commandes;
{ Pré : 1 <= num_bloc <= lire_nbre_blocs_scene (nom_fich) }
{ Post : renvoie le bloc de commandes numéro num_bloc de la scène dont le nom de fichier est nom_fich }

 procedure ecrire_bloc_scene_nr (nom_fich: str32;
           num_bloc: integer;
           var bloc_com: Tbloc_commandes);
{ Pré : 1 <= num_bloc <= lire_nbre_blocs_scene (nom_fich) + 1 }
{ Post : si num_bloc <= lire_nbre_blocs_scene (nom_fich), le bloc de commandes numéro num_bloc }
{       de la scène dont le nom de fichier est nom_fich est remplacé par bloc_com sinon bloc_com }
{       est ajouté a la fin du fichier nom_fich }

 procedure effacer_a_partir_bloc_scene_nr (nom_fich: str32;
           num_bloc: integer);
{ Pré : 1 <= num_bloc <= lire_nbre_blocs_scene (nom_fich) }
{ Post : tous les blocs de commandes dont le numéro est compris entre num_bloc }
{       et lire_nbre_blocs_scene (nom_fich) sont retirés du fichier nom_fich }

 procedure nouv_fich_trace (nom_enfant: str255;
           var nom_fich: str255);
{ Pré : / }
{ Post : un nouveau fichier de trace est créé pour l'utilisateur nom_enfant, }
{       le nom unique de ce fichier est retourné dans nom_fich }

 procedure ecrire_bloc_trace (nom_fich: str255;
           var bloc_lignes: Tbloc_trace;
           nbre_lignes: integer);
{ Pré : nom_fich identifie un fichier de trace, 1 <= nbre_lignes <= nbre_lignes_bloc_tr }
{ les nbre_lignes premières lignes de bloc_lignes sont écrites à la fin du fichier nom_fich }

implementation

 var
  fich_res_appl: integer;
  fich_res_deco: integer;
  fich_res_pers: integer;
  fich_res_musi: integer;
  erreur: OSErr;
  nom_vol: StringPtr;
  vol_ref_num: integer;
  fich_num: integer;


 procedure nouv_param_defaut (var param: Tparam_enfant); {procédure locale}
```

```
 var
  i: integer;

begin
 with param do
  begin
   nom_enf := '';
   mode_util := 1;
   vitesse_def := 90;
   type_trace := 1;
   enreg_autom := false;
   for i := 1 to 3 do
    begin
     commandes[i, 1] := 15;
     commandes[i, 2] := 16;
    end;
   commandes[1, 3] := 26;
   commandes[1, 4] := 17;
   commandes[1, 5] := 3;
   commandes[1, 6] := 4;
   commandes[1, 7] := 18;
   commandes[1, 8] := 19;
   commandes[2, 3] := 20;
   commandes[2, 4] := 21;
   commandes[2, 5] := 22;
   commandes[2, 6] := 25;
   commandes[2, 7] := 23;
   commandes[2, 8] := 24;
   commandes[3, 3] := 10;
   commandes[3, 4] := 7;
   for i := 5 to max_com do
    commandes[3, i] := -1;
   for i := 1 to max_com do
    commandes[4, i] := -1;
  end;
end; {nouv_param_defaut}


function chercher_fichier (nom_fich: str255;
          type_fich: OSType): integer;

 var
  num_fich: integer;
  reponse: SFReply;
  tab_types: SFTypeList;
  point_dial: point;
  vol_ref_handle: handle;
  vol_ref_fich: integer;
  continue_recherche: boolean;

begin
 num_fich := OpenResFile(nom_fich);
 if num_fich = -1 then
  begin
   vol_ref_handle := GetNamedResource('VoRN', nom_fich);
   if vol_ref_handle <> nil then
    begin
     HLock(vol_ref_handle);
```

```
        BlockMove(vol_ref_handle^, @vol_ref_fich, 2);
       end
      else
       vol_ref_fich := 0;
      num_fich := OpenRFPerm(nom_fich, vol_ref_fich, 0);
      if num_fich = -1 then
       continue_recherche := erreur_ouverture_fichier(nom_fich)
      else
       continue_recherche := false;
      if continue_recherche = true then
       begin
        SetPt(point_dial, 50, 50);
        tab_types[0] := type_fich;
        SFGetFile(point_dial, '', nil, 1, tab_types, nil, reponse);
        if reponse.good = true then
         begin
          num_fich := OpenRFPerm(reponse.FName, reponse.VRefNum, 0);
          BlockMove(@reponse.VRefNum, vol_ref_handle^, 2);
          ChangedResource(vol_ref_handle);
          WriteResource(vol_ref_handle);
         end
        else
         num_fich := -1;
       end;
      HUnlock(vol_ref_handle);
      ReleaseResource(vol_ref_handle);
     end;
   chercher_fichier := num_fich;
end;


procedure ouvrir_fichiers;

 var
  param_defaut: Tparam_enfant;
  nbre_bytes: Longint;
  chercher_fich: boolean;

begin
 erreur := GetVol(nom_vol, vol_ref_num);
 erreur := FSOpen('Parametres.Enf', vol_ref_num, fich_num);
 if erreur <> 0 then
  begin
   if erreur = FnFErr then
    begin
     erreur := create('Parametres.Enf', vol_ref_num, 'MXLO', 'PARA');
     if erreur <> NoErr then
      erreur_ecriture_fichier('Parametres.Enf', erreur)
     else
      begin
       nouv_param_defaut(param_defaut);
       nbre_bytes := SizeOf(param_defaut);
       erreur := FSOpen('Parametres.Enf', vol_ref_num, fich_num);
       erreur := FSWrite(fich_num, nbre_bytes, @param_defaut);
       if erreur <> NoErr then
        erreur_ecriture_fichier('Parametres.Enf', erreur);
      end
    end
   else
```

```
      erreur_ecriture_fichier('Parametres.Enf', erreur);
    end;
  erreur := FSClose(fich_num);
  fich_res_appl := CurResFile;
  fich_res_deco := chercher_fichier('Decors.Res', 'DECO');
  fich_res_pers := chercher_fichier('Personnages.Res', 'PERS');
  fich_res_musi := chercher_fichier('Musiques.Res', 'MUSI');
end; {ouvrir_fichiers}

procedure fermer_fichiers;

begin
  CloseResFile(fich_res_deco);
  CloseResFile(fich_res_pers);
  CloseResFile(fich_res_musi);
  erreur := FlushVol(nil, vol_ref_num);
end; {fermer_fichiers}

procedure ecrire_parametres_enfant;

  var
    param_temp : Tparam_enfant;
    nbre_bytes : LongInt;
    trouve : boolean;
    pos : LongInt;
    fin_fich : LongInt;

begin
  UprString(parametres.nom_enf, false);
  trouve := false;
  pos := 0;
  nbre_bytes := SizeOf(param_temp);
  erreur := FSOpen('Parametres.Enf', vol_ref_num, fich_num);
  erreur := GetEoF(fich_num, fin_fich);
  erreur := SetFPos(fich_num, FSFromStart, pos);
  while (pos < fin_fich) and (trouve = false) do
    begin
      erreur := FSRead(fich_num, nbre_bytes, @param_temp);
      pos := pos + nbre_bytes;
      if param_temp.nom_enf = parametres.nom_enf then
        trouve := true;
    end;
  if trouve = true then
    begin
      pos := pos - nbre_bytes;
      erreur := SetFPos(fich_num, FSFromStart, pos);
    end;
  erreur := FSWrite(fich_num, nbre_bytes, @parametres);
  if erreur <> NoErr then
    erreur_ecriture_fichier('Paramètres.Enf', erreur);
  erreur := FSClose(fich_num);
end; {ecrire_parametres_enfant}

function lire_parametres_enfant;

  var
    param_temp : Tparam_enfant;
    trouve : boolean;
    pos, fin_fich : Longint;
```

```
    nbre_bytes: LongInt;

begin
  erreur := FSOpen('Parametres.Enf', vol_ref_num, fich_num);
  if erreur = Noerr then
    begin
      UprString(nom_enfant, false);
      trouve := false;
      pos := 0;
      erreur := GetEoF(fich_num, fin_fich);
      while (pos < fin_fich) and (trouve = false) do
        begin
          nbre_bytes := SizeOf(param_temp);
          erreur := FSRead(fich_num, nbre_bytes, @param_temp);
          if param_temp.nom_enf = nom_enfant then
            trouve := true;
          pos := pos + nbre_bytes;
        end;
      if trouve = false then
        begin
          nbre_bytes := SizeOf(param_temp);
          erreur := SetFPos(fich_num, FSFromStart, 0);
          erreur := FSRead(fich_num, nbre_bytes, @param_temp);
          param_temp.nom_enf := nom_enfant;
        end;
      erreur := FSClose(fich_num);
    end
  else
    nouv_param_defaut(param_temp);
  lire_parametres_enfant := param_temp;
end; {lire_parametres_enfant}

function lire_illustration_nr;

  var
    image: PicHandle;

begin
  UseResFile(fich_res_appl);
  image := GetPicture(12999 + numero);
  HNoPurge(Handle(image));
  lire_illustration_nr := image;
end; {lire_illustration_nr}

function lire_nbre_decors;

begin
  UseResFile(fich_res_deco);
  lire_nbre_decors := Count1Resources('PICT');
end; {lire_nbre_decors}

function lire_decor_nr;

  var
    image: PicHandle;

begin
  UseResFile(fich_res_deco);
  image := GetPicture(9999 + numero);
```

```
  HNoPurge(Handle(image));
  lire_decor_nr := image;
end; {lire_decor_nr}


function lire_nbre_personnages;

  var
    nombre_images, reste : integer;
begin
  UseResFile(fich_res_pers);
  nombre_images := Count1Resources('PICT');
  reste := nombre_images mod 19;
  if reste = 0 then
    lire_nbre_personnages := nombre_images div 19
  else
    lire_nbre_personnages := (nombre_images div 19) + 1;
end; {lire_nbre_personnages}

function lire_personnage_nr_att;

  var
    num_res : integer;
    image : PicHandle;
    rect1 : rect;

begin
  UseResFile(fich_res_pers);
  num_res := (100 * numero) + attitude - 1;
  image := GetPicture(num_res);
  HNoPurge(Handle(image));
  if image = nil then
    erreur_attitude_manquante(numero, attitude);
  lire_personnage_nr_att := image;
end; {lire_personnage_nr_att}

function lire_ecran_fixe_nr;

  var
    image : PicHandle;

begin
  UseResFile(fich_res_appl);
  image := GetPicture(11999 + numero);
  HNoPurge(Handle(image));
  lire_ecran_fixe_nr := image;
end; {lire_ecran_fixe_nr}

function lire_nbre_musiques;

begin
  UseResFile(fich_res_musi);
  lire_nbre_musiques := Count1Resources('snd ');
end; {lire_nbre_musiques}

function lire_ecran_musique_nr;

  var
    image : pichandle;
```

```
begin
 UseResFile(fich_res_musi);
 image := GetPicture(10999 + numero);
 HNoPurge(Handle(image));
 lire_ecran_musique_nr := image;
end; {lire_ecran_musique_nr}

function lire_musique_nr;

 var
  musique : Handle;

begin
 UseResFile(fich_res_musi);
 musique := GetResource('snd ', 10999 + numero);
 HNoPurge(musique);
 lire_musique_nr := musique;
end; {lire_musique_nr}

function lire_nbre_scenes_enf;

 var
  nom_fich_ref : str255;
  fin_fich : Longint;

begin
 if nom_enfant <> '' then
  begin
   UprString(nom_enfant, true);
   nom_enfant := Concat(nom_enfant, '.');
  end;
 nom_fich_ref := Concat(nom_enfant, 'Scenes.Ref');
 erreur := FSOpen(nom_fich_ref, vol_ref_num, fich_num);
 if erreur <> 0 then
  lire_nbre_scenes_enf := 0
 else
  begin
   erreur := GetEoF(fich_num, fin_fich);
   erreur := FSClose(fich_num);
   lire_nbre_scenes_enf := fin_fich div SizeOf(TScene_ref);
  end;
end; {lire_nbre_scenes}

function lire_reference_scene_enf_nr;

 var
  nom_fich_ref : str255;
  reference : Tscene_ref;
  nom_fich : str255;
  pos, nbre_bytes : LongInt;

begin
 if nom_enfant <> '' then
  begin
   UprString(nom_enfant, true);
   nom_enfant := Concat(nom_enfant, '.');
  end;
 nom_fich_ref := Concat(nom_enfant, 'Scenes.Ref');
```

```
  erreur := FSOpen(nom_fich_ref, vol_ref_num, fich_num);
  pos := LongInt(num_ref - 1) * SizeOf(reference);
  erreur := SetFPos(fich_num, FSFromStart, pos);
  nbre_bytes := SizeOf(reference);
  erreur := FSRead(fich_num, nbre_bytes, @reference);
  lire_reference_scene_enf_nr := reference;
  erreur := FSClose(fich_num);
end; {lire_reference_scene_nr}

procedure nouv_fich_res_ref (nom_fich: str255;
          var fich_res: integer);

 var
  num: LongInt;
  num_string: str255;
  string_h: StringHandle;

begin
 num := 0;
 NumToString(num, num_string);
 CreateResFile(nom_fich);
 fich_res := OpenResFile(nom_fich);
 string_h := StringHandle(NewHandle(256));
 string_h^^ := num_string;
 AddResource(Handle(string_h), 'STR ', UniqueId('STR '), 'num_der_scene');
 AddResource(Handle(string_h), 'STR ', UniqueId('STR '), 'num_der_trace');
 UpdateResFile(fich_res);
 DisposHandle(handle(string_h));
end;

procedure ecrire_reference_scene_enf_nr;

 var
  fich_res_ref_scenes: integer;
  nom_fich_ref: str255;
  num_der_scene: LongInt;
  num_der_scene_str: str255;
  num_der_scene_hdl: handle;
  str_hdl: StringHandle;
  ref_temp: Tscene_ref;
  nbre_bytes: LongInt;
  pos: LongInt;
  fin_fich: longint;

begin
 if nom_enfant <> '' then
  begin
   UprString(nom_enfant, true);
   nom_enfant := Concat(nom_enfant, '.');
  end;
 nom_fich_ref := Concat(nom_enfant, 'Scenes.Ref');
 erreur := FSOpen(nom_fich_ref, vol_ref_num, fich_num);
 if erreur <> NoErr then
  begin
   if erreur = FnFErr then
    begin
     erreur := Create(nom_fich_ref, vol_ref_num, 'MXLO', 'SREF');
     if erreur <> NoErr then
      erreur_ecriture_fichier(nom_fich_ref, erreur)
```

```
      else
       begin
        fich_res_ref_scenes := OpenResFile(nom_fich_ref);
        if fich_res_ref_scenes = -1 then
         nouv_fich_res_ref(nom_fich_ref, fich_res_ref_scenes);
        CloseResFile(fich_res_ref_scenes);
        erreur := FSOpen(nom_fich_ref, vol_ref_num, fich_num);
       end;
     end
    else
     erreur_ecriture_fichier(nom_fich_ref, erreur);
  end;
 if erreur = NoErr then
  begin
   erreur := GetEoF(fich_num, fin_fich);
   pos := LongInt(num_ref - 1) * SizeOf(reference);
   if pos = fin_fich then
    begin
     erreur := FSClose(fich_num);
     fich_res_ref_scenes := OpenResFile(nom_fich_ref);
     UseResFile(fich_res_ref_scenes);
     num_der_scene_hdl := GetNamedResource('STR ', 'num_der_scene');
     HNoPurge(num_der_scene_hdl);
     str_hdl := StringHandle(num_der_scene_hdl);
     num_der_scene_str := str_hdl^^;
     StringToNum(num_der_scene_str, num_der_scene);
     num_der_scene := num_der_scene + 1;
     NumToString(num_der_scene, num_der_scene_str);
     str_hdl^^ := num_der_scene_str;
     num_der_scene_hdl := handle(str_hdl);
     ChangedResource(num_der_scene_hdl);
     WriteResource(num_der_scene_hdl);
     HPurge(num_der_scene_hdl);
     reference.nom_fich := Concat(nom_enfant, 'SceneEnreg.');
     reference.nom_fich := Concat(reference.nom_fich, num_der_scene_str);
     CloseResFile(fich_res_ref_scenes);
     DisposHandle(Handle(str_hdl));
     erreur := FSOpen(nom_fich_ref, vol_ref_num, fich_num);
     erreur := SetFPos(fich_num, FSFromStart, pos);
    end
   else
    begin
     nbre_bytes := SizeOf(ref_temp);
     erreur := SetFPos(fich_num, FSFromStart, pos);
     erreur := FSRead(fich_num, nbre_bytes, @ref_temp);
     reference.nom_fich := ref_temp.nom_fich;
     erreur := SetFPos(fich_num, FSFromStart, pos);
    end;
   nbre_bytes := SizeOf(reference);
   erreur := FSWrite(fich_num, nbre_bytes, @reference);
   if erreur <> NoErr then
    erreur_ecriture_fichier(nom_fich_ref, erreur);
  end;
 erreur := FSClose(fich_num);
end; {ecrire_reference_scene_nr}

procedure effacer_scene_enf_nr;

 var
```

```
    nom_fich_ref: str255;
    reference: Tscene_ref;
    fin_fich, pos, pos_remplace, nbre_bytes: LongInt;

begin
 if nom_enfant <> '' then
  begin
   UprString(nom_enfant, true);
   nom_enfant := Concat(nom_enfant, ':');
  end;
 nom_fich_ref := Concat(nom_enfant, 'Scenes.Ref');
 erreur := FSOpen(nom_fich_ref, vol_ref_num, fich_num);
 nbre_bytes := SizeOf(reference);
 pos := LongInt(num_ref - 1) * nbre_bytes;
 erreur := SetFPos(fich_num, FSFromStart, pos);
 erreur := FSRead(fich_num, nbre_bytes, @reference);
 erreur := FSDelete(reference.nom_fich, vol_ref_num);
 if erreur <> NoErr then
  erreur_ecriture_fichier(reference.nom_fich, erreur);
 erreur := GetEoF(fich_num, fin_fich);
 pos_remplace := pos;
 pos := pos + nbre_bytes;
 while (pos < fin_fich) and (erreur = NoErr) do
  begin
   nbre_bytes := SizeOf(reference);
   erreur := FSRead(fich_num, nbre_bytes, @reference);
   pos := pos + nbre_bytes;
   erreur := SetFPos(fich_num, FSFromStart, pos_remplace);
   erreur := FSWrite(fich_num, nbre_bytes, @reference);
   if erreur <> NoErr then
    erreur_ecriture_fichier(nom_fich_ref, erreur);
   pos_remplace := pos_remplace + nbre_bytes;
   erreur := SetFPos(fich_num, FSFromStart, pos);
  end;
 erreur := SetEof(fich_num, pos_remplace);
 if erreur <> NoErr then
  erreur_ecriture_fichier(nom_fich_ref, erreur);
 erreur := FSClose(fich_num);
end; {effacer_scene_nr}

function lire_nbre_blocs_scene;

 var
  fin_fich: LongInt;

begin
 erreur := FSOpen(nom_fich, vol_ref_num, fich_num);
 if erreur <> 0 then
  lire_nbre_blocs_scene := 0
 else
  begin
   erreur := GetEoF(fich_num, fin_fich);
   erreur := FSClose(fich_num);
   lire_nbre_blocs_scene := fin_fich div SizeOf(Tbloc_commandes);
  end;
end; {lire_nbre_blocs_scene}

function lire_bloc_scene_nr;
```

```
  var
   bloc_com : Tbloc_commandes;
   pos, nbre_bytes : LongInt;

begin
 nbre_bytes := SizeOf(bloc_com);
 pos := LongInt(num_bloc - 1) * nbre_bytes;
 erreur := FSOpen(nom_fich, vol_ref_num, fich_num);
 erreur := SetFPos(fich_num, FSFromStart, pos);
 erreur := FSRead(fich_num, nbre_bytes, @bloc_com);
 lire_bloc_scene_nr := bloc_com;
 erreur := FSClose(fich_num);
end; {lire_bloc_scene_nr}

procedure ecrire_bloc_scene_nr;

  var
   nbre_bytes : LongInt;
   pos : LongInt;

begin
 erreur := FSOpen(nom_fich, vol_ref_num, fich_num);
 if erreur <> NoErr then
  begin
   if erreur = FnFErr then
    erreur := Create(nom_fich, vol_ref_num, 'MXLO', 'SCEN');
   if erreur <> NoErr then
    erreur_ecriture_fichier(nom_fich, erreur)
   else
    erreur := FSOpen(nom_fich, vol_ref_num, fich_num);
  end;
 if erreur = NoErr then
  begin
   nbre_bytes := SizeOf(bloc_com);
   pos := LongInt(num_bloc - 1) * nbre_bytes;
   erreur := SetFPos(fich_num, FSFromStart, pos);
   erreur := FSWrite(fich_num, nbre_bytes, @bloc_com);
   if erreur <> NoErr then
    erreur_ecriture_fichier(nom_fich, erreur);
  end;
 erreur := FSClose(fich_num);
end; {ecrire_bloc_scene_nr}

procedure effacer_a_partir_bloc_scene_nr;

  var
   nouv_fin_fich : LongInt;

begin
 nouv_fin_fich := (num_bloc - 1) * SizeOf(Tbloc_commandes);
 erreur := FSOpen(nom_fich, vol_ref_num, fich_num);
 erreur := SetEof(fich_num, nouv_fin_fich);
 if erreur <> NoErr then
  erreur_ecriture_fichier(nom_fich, erreur);
 erreur := FSClose(fich_num);
end; {effacer_a_partir_bloc_scene_nr}

procedure nouv_fich_trace;
```

```
  var
    nom_fich_ref: str255;
    num_der_trace_str: str255;
    num_der_trace_hdl: Handle;
    num_der_trace: LongInt;
    str_hdl: StringHandle;
    fich_res_ref_scenes: integer;

begin
  if nom_enfant <> '' then
    begin
      UprString(nom_enfant, true);
      nom_enfant := Concat(nom_enfant, '.');
    end;
  nom_fich_ref := Concat(nom_enfant, 'Scenes.Ref');
  fich_res_ref_scenes := OpenResFile(nom_fich_ref);
  if fich_res_ref_scenes = -1 then
    nouv_fich_res_ref(nom_fich_ref, fich_res_ref_scenes);
  UseResFile(fich_res_ref_scenes);
  num_der_trace_hdl := GetNamedResource('STR ', 'num_der_trace');
  if num_der_trace_hdl = nil then
    begin
      NumToString(1, num_der_trace_str);
      str_hdl := StringHandle(NewHandle(256));
      str_hdl^^ := num_der_trace_str;
      AddResource(Handle(str_hdl), 'STR ', UniqueId('STR '), 'num_der_trace');
      UpdateResFile(fich_res_ref_scenes);
    end
  else
    begin
      HNoPurge(num_der_trace_hdl);
      str_hdl := StringHandle(num_der_trace_hdl);
      num_der_trace_str := str_hdl^^;
      StringToNum(num_der_trace_str, num_der_trace);
      num_der_trace := num_der_trace + 1;
      NumToString(num_der_trace, num_der_trace_str);
      str_hdl^^ := num_der_trace_str;
      num_der_trace_hdl := handle(str_hdl);
      ChangedResource(num_der_trace_hdl);
      WriteResource(num_der_trace_hdl);
      HPurge(num_der_trace_hdl);
    end;
  nom_fich := Concat(nom_enfant, 'trace.');
  nom_fich := Concat(nom_fich, num_der_trace_str);
  erreur := create(nom_fich, vol_ref_num, 'MACA', 'TEXT');
  if erreur <> NoErr then
    erreur_ecriture_fichier(nom_fich, erreur);
  CloseResFile(fich_res_ref_scenes);
  DisposHandle(Handle(str_hdl));
end;

procedure ecrire_bloc_trace;

  var
    nbre_bytes: LongInt;

begin
  nbre_bytes := nbre_lignes * SizeOf(Tligne_trace);
  erreur := FSOpen(nom_fich, vol_ref_num, fich_num);
```

```
    erreur := SetFPos(fich_num, FsFromLEOF, 0);
    erreur := FSWrite(fich_num, nbre_bytes, @bloc_lignes);
    if erreur <> NoErr then
      erreur_ecriture_fichier(nom_fich, erreur);
    erreur := FSClose(fich_num);
  end; {ecrire_bloc_trace_nr}

end. {unit fichiers}
```

```
unit musique;

interface

 uses
  Sound, messages_erreurs;

 var
  joue_musique : boolean; { vaut TRUE lorsqu'un thème musical est en train de se jouer }

 procedure init_musique;
{ Pré : / }
{ Post : les autres procédures du module peuvent être utilisées }

 procedure charger_musique (nouv_mus : handle);
{ Pré : nouv_mus est un handle vers une ressource de type "snd " ou vaut nil }
{ Post : nouv_mus est prête à être jouée; toute musique qui aurait été précédemment chargée }
{        en mémoire par un appel à charger_musique est supprimée de la mémoire }

 procedure demarrer_musique;
{ Pré : / }
{ Post : si une musique a été précédemment chargée en mémoire par un appel à charger_musique, }
{        elle se joue en processus de fond }

 procedure verifier_musique;
{ Pré : / }
{ Post : appelée régulièrement, cette procédure assure la continuité de la musique }

 procedure terminer_musique;
{ Pré : / }
{ Post : la musique s'arrête }

 procedure jeter_musique;
{ Pré : / }
{ Post : il n'y a plus de musique chargée en mémoire }

implementation

 var
  canal: SndChannelPtr;
  debut_musique: Ptr;
  syst_mus_ok: boolean;
  musique: Handle;


 function OffsetPtr (BasePtr: Ptr;
           Offset: integer): Ptr;

  var
   LongIntPtr: LongInt;

 begin
  LongIntPtr := LongInt(BasePtr);
  OffsetPtr := Ptr(LongIntPtr + LongInt(offset));
 end; {OffsetPtr}

 procedure init_musique;

  var
```

```
    version_systeme : handle;
    num_vers, num_impl : SignedByte;
    debut_mot : Ptr;

begin
  version_systeme := GetResource('vers', 1);
  HNoPurge(version_systeme);
  HLock(version_systeme);
  debut_mot := version_systeme^;
  BlockMove(debut_mot, @num_vers, 1);
  debut_mot := OffsetPtr(debut_mot, 1);
  BlockMove(debut_mot, @num_impl, 1);
  HUnlock(version_systeme);
  HPurge(version_systeme);
  ReleaseResource(version_systeme);
  if (num_vers < 6) or ((num_vers >= 6) and (num_impl < 2)) then
    begin
      erreur_systeme_musique;
      syst_mus_ok := false;
    end
  else
    syst_mus_ok := true;
  joue_musique := false;
  canal := nil;
  musique := nil;
  debut_musique := nil;
end; {init_musique}

procedure charger_musique;

begin
  if syst_mus_ok = true then
    begin
      if musique <> nil then
        jeter_musique;
      musique := nouv_mus;
    end;
end; {charger_musique}

procedure demarrer_musique;

  var
    pointeur_temp : Ptr;
    erreur : OSErr;
    nbre_synthes, nbre_com : integer;
    SM_commande : SndCommand;

begin
  if (musique <> nil) and (joue_musique = false) then
    begin
      HLock(musique);
      pointeur_temp := OffsetPtr(musique^, 2);
      BlockMove(pointeur_temp, @nbre_synthes, 2);
      pointeur_temp := OffsetPtr(pointeur_temp, nbre_synthes * 6 + 2);
      BlockMove(pointeur_temp, @nbre_com, 2);
      debut_musique := OffsetPtr(pointeur_temp, nbre_com * 8 + 2);
      canal := nil;
      erreur := SndNewChannel(canal, 5, InitMono, nil);
      with SM_commande do
```

```
      begin
       Cmd := AmpCmd;
       param1 := 255;
       param2 := 0;
      end;
     erreur := SndDoCommand(canal, SM_commande, false);
     with SM_commande do
      begin
       Cmd := RateCmd;
       param1 := 0;
       param2 := 1;
      end;
     erreur := SndDoCommand(canal, SM_commande, false);
     with SM_commande do
      begin
       Cmd := BufferCmd;
       param1 := 0;
       param2 := LongInt(debut_musique);
      end;
     erreur := SndDoCommand(canal, SM_commande, false);
     joue_musique := true;
   end;
end;

procedure terminer_musique;

 var
  erreur : OSErr;

begin
 if joue_musique = true then
  begin
   HUnlock(musique);
   erreur := SndDisposeChannel(canal, true);
   joue_musique := false;
   canal := nil;
   debut_musique := nil;
  end;
end;

procedure verifier_musique;

 var
  SM_commande : SndCommand;
  erreur : OSErr;

begin
 if joue_musique = true then
  begin
   with SM_commande do
    begin
     Cmd := BufferCmd;
     param1 := 0;
     param2 := LongInt(debut_musique);
    end;
   erreur := SndDoCommand(canal, SM_commande, true);
  end;
end;
```

```
 procedure jeter_musique;

 begin
  terminer_musique;
  HPurge(musique);
  ReleaseResource(musique);
  musique := nil;
 end;
end. {unit musique}
```

```
unit animation;

interface

 uses
  fichiers;

 const
  max_attitudes = 18; { nombre maximum d'attitudes (dessins) pour un personnage, la tête seule non comprise }
  max_personnages = 4; { nombre maximum de personnages pouvant être utilisés en même temps }
  taille_deplacement = 28; { longueur totale, en points, d'un déplacement de marionnette }
  taille_phase_deplacement = 4; { distance, en point, entre deux affichages lors d'un déplacement }

 type
  Tpersonnage = record { structure décrivant un personnage }
     pers_pos : rect; { rectangle définissant la position du personnage }
     pers_visible : boolean; { TRUE si le personnage doit être affiché à l'écran }
     pers_attitude : integer; { numéro de l'attitude courante du personnage }
     tab_attitudes : array [1..max_attitudes] of bitmap;
{ tableau contenant les différents dessins sous forme de bitmaps }
     tab_mask : array [1..max_attitudes] of bitmap;
{ masques permettant d'éviter que les personnages n'apparaissent dans un rectangle }
     tete : PicHandle; { tête seule, de type PICT }
     num_pers_disque : integer; { identifiant du personnage dans la B.D. (module fichiers) }
   end;

 var
  tab_pers : array [1..max_personnages] of Tpersonnage; { tableau contenant les personnages à animer }
  tab_ordre_pers : array [1..max_personnages] of integer;
{ tableau indiquant l'ordre dans lequel les personnages doivent être affichés }
  nbre_pers_crees : integer; { nombre de personnages placés en mémoire pour être animés }
  decor : bitmap; { décor à utiliser dans l'animation }
  rectangle_limite : rect; { rectangle délimitant la scène }
  port_memoire : GrafPtr; { port graphique en mémoire centrale servant à la reconstitution des images }
  ecran_scene : WindowPtr; { fenêtre contenant la scène }
  fenetre_lapin : WindowPtr; { fenêtre destinée à recevoir le lapin indiquant un temps d'attente }


 procedure init_animation;
{ Pré : / }
{ Post : les initialisations nécessaires à l'utilisation des autres routines du module ont été effectuées }

 procedure retrace_ecran;
{ Pré : / }
{ Post : l'écran retracé avec le décor et les personnages devant s'y trouver }

 procedure laisser_place_barre_menu;
{ Pré : / }
{ Post : les 20 lignes du haut de l'écran sont nettoyées et réservées aux menus déroulants }

 procedure non_laisser_place_barre_menu;
{ Pré : / }
{ Post : les 20 lignes du haut de l'écran sont recouvertes par la scène }

 procedure cree_pers_face (nr_pers_tab, nr_pers_disk : integer);
{ Pré : 1 <= nr_pers_tab <= nbre_pers_crees + 1 ; nr_pers_tab <= max_personnages }
{       1 <= nr_pers_disk <= lire_nbre_personnages (voir unit fichiers) }
{ Post : si nr_pers_tab <= nbre_pers_crees, le personnage contenu dans tab_pers [nr_pers_tab] }
{        est remplacé par celui dont l'identifiant dans la B.D. est nr_pers_disk. }
```

```
{       si nr_pers_tab = nbre_pers_crees, alors nbre_pers_crees := nbre_pers_crees + 1 }
{       et tab_pers [nbre_pers_crees] contient le personnage identifié dans la B.D. par nr_pers_disk. }
{       Dans les deux cas, seule l'attitude "debout de face" du personnage est chargée en mémoire }

 procedure cree_pers_complet (nr_pers_tab, nr_pers_disk: integer);
{ Spécifications identiques que cree_pers_face, mais l'ensemble des attitudes du personnage est chargé en
      mémoire }

 procedure detruit_personnage (nr_pers_tab: integer);
{ Pré : 1 <= nr_pers_tab <= nbre_pers_crees }
{ Post : la mémoire utilisée par le personnage contenu dans tab_pers [nr_pers_tab] est libérée; }
{       tous les personnages de tab_pers dont l'indice est supérieur à nr_pers_tab sont décalés }
{       vers la gauche du tableau et nbre_pers_crees := nbre_pers_crees - 1 }

 procedure place_decor (pic_decor: PicHandle);
{ Pré : / }
{ Post : l'image spécifiée par pic_décor est stockée dans decor sous forme de bitmap et est }
{       affichée à l'écran. Elle sera utilisée comme toile de fond pour toutes les fonctions d'animation }

 procedure place_personnage (nr_pers, nr_attit, ligne, colonne: integer);
{ Pré : 1 <= nr_pers <= nbre_pers_crees; 1 <= nr_attit <= 18 }
{ Post : le personnage contenu dans tab_pers (nr_pers) est positionné en (ligne,colonne), }
{       dans l'attitude nr_attit }

 procedure entrer_scene (nr_pers: integer);
{ Pré : 1 <= nr_pers <= nbre_pers_crees }
{ Post : si le personnage contenu dans tab_pers [nr_pers] n'est pas affiché à l'écran, }
{       il entre en scène par le bas de l'écran }

 procedure sortir_scene (nr_pers: integer);
{ Pré : 1 <= nr_pers <= nbre_pers_crees }
{ Post : si le personnage contenu dans tab_pers [nr_pers] est affiché à l'écran, }
{       il sort de scène par le bas de l'écran }

 procedure efface_personnage (nr_pers: integer);
{ Pré : 1 <= nr_pers <= nbre_pers_crees }
{ Post : si le personnage contenu dans tab_pers [nr_pers] est affiché à l'écran, il disparaît }

 procedure deplace_gauche (nr_pers: integer);
{ Pré : 1 <= nr_pers <= nbre_pers_crees }
{ Post : si le personnage contenu dans tab_pers [nr_pers] est affiché à l'écran, }
{       il se déplace de taille_déplacement pixels vers la gauche. }
{       Le déplacement s'effectue en marchant si le personnage est debout, en "glissant" sinon }

 procedure deplace_droite (nr_pers: integer);
{ Pré : 1 <= nr_pers <= nbre_pers_crees }
{ Post : si le personnage contenu dans tab_pers [nr_pers] est affiché à l'écran, }
{       il se déplace de taille_déplacement pixels vers la droite. }
{       Le déplacement s'effectue en marchant si le personnage est debout, en "glissant" sinon }

 procedure deplace_haut (nr_pers: integer);
{ Pré : 1 <= nr_pers <= nbre_pers_crees }
{ Post : si le personnage contenu dans tab_pers [nr_pers] est affiché à l'écran, }
{       il se déplace en "glissant" de taille_déplacement pixels vers le haut. }

 procedure deplace_bas (nr_pers: integer);
{ Pré : 1 <= nr_pers <= nbre_pers_crees }
{ Post : si le personnage contenu dans tab_pers [nr_pers] est affiché à l'écran, }
{       il se déplace en "glissant" de taille_déplacement pixels vers le bas. }
```

```
 procedure tourne_gauche (nr_pers: integer);
{ Pré : 1 <= nr_pers <= nbre_pers_crees }
{Post : si le personnage contenu dans tab_pers [nr_pers] est affiché à l'écran, }
{       il pivote d'un quart de tour vers la gauche. }

 procedure tourne_droite (nr_pers: integer);
{ Pré : 1 <= nr_pers <= nbre_pers_crees }
{Post : si le personnage contenu dans tab_pers [nr_pers] est affiché à l'écran, }
{       il pivote d'un quart de tour vers la droite. }

 procedure lever (nr_pers: integer);
{ Pré : 1 <= nr_pers <= nbre_pers_crees }
{Post : si le personnage contenu dans tab_pers [nr_pers] est affiché à l'écran, }
{       il est placé en position debout. }

 procedure asseoir (nr_pers: integer);
{ Pré : 1 <= nr_pers <= nbre_pers_crees }
{Post : si le personnage contenu dans tab_pers [nr_pers] est affiché à l'écran, }
{       il est placé en position assise. }


 procedure coucher (nr_pers: integer);
{ Pré : 1 <= nr_pers <= nbre_pers_crees }
{Post : si le personnage contenu dans tab_pers [nr_pers] est affiché à l'écran, }
{       il est placé en position couchée. }


 procedure agiter (nr_pers: integer);
{ Pré : 1 <= nr_pers <= nbre_pers_crees }
{Post : si le personnage contenu dans tab_pers [nr_pers] est affiché à l'écran, il agite les bras. }


 procedure montre_lapin;
{ Pré : / }
{ Post : une fenêtre contenant un lapin est affichée au centre de l'écran }

 procedure change_lapin;
{ Pré : / }
{ Post : si la fenêtre contenant le lapin est à l'écran, }
{       la position de la patte et des oreilles du lapin a changé }

 procedure cache_lapin;
{ Pré : / }
{ Post : la fenêtre contenant le lapin n'est plus affichée à l'écran }

implementation

 var
  num_lapin: integer;
  lapin1, lapin2: BitMap;
  rect_lapin: rect;
  menu_rgn: RgnHandle;

 procedure init_animation;

  var
   port_sauvegarde: GrafPtr;
```

```
    nbre_colonnes : integer;
    taille_ecran_animation : LongInt;
    rectangle_fenetre_animation : rect;
    adresse_port_memoire : Ptr;
    bitmap_port_memoire : BitMap;
    adresse_decor : Ptr;
    rect_menu : rect;
    pic_lapin1 , pic_lapin2 : PicHandle;
    rect_lapin1 , rect_lapin2 : rect;
    rectangle_fenetre_lapin : rect;
    larg_rect, haut_rect : integer;

begin
  GetPort(port_sauvegarde);
  nbre_pers_crees := 0;
  with rectangle_limite do
    begin
      left := 0;
      top := 0;
      right := 512;
      bottom := 282;
    end;
  rectangle_fenetre_animation := rectangle_limite;
  OffsetRect(rectangle_fenetre_animation, (ScreenBits.Bounds.Right - 512) div 2, (ScreenBits.Bounds.Bottom -
      342) div 2);
  ecran_scene := NewWindow(nil, rectangle_fenetre_animation, '', true, 2, WindowPtr(-1), false, 0);
  ShowWindow(ecran_scene);
  nbre_colonnes := (ScreenBits.Bounds.Right - ScreenBits.Bounds.Left) div 8;
  taille_ecran_animation := nbre_colonnes * LongInt(rectangle_limite.bottom - rectangle_limite.top);
  adresse_port_memoire := NewPtr(taille_ecran_animation);
  with bitmap_port_memoire do
    begin
      BaseAddr := adresse_port_memoire;
      RowBytes := nbre_colonnes;
      Bounds := rectangle_limite;
    end;
  New(port_memoire);
  OpenPort(port_memoire);
  SetPort(port_memoire);
  SetPortBits(bitmap_port_memoire);
  adresse_decor := NewPtr(taille_ecran_animation);
  with decor do
    begin
      BaseAddr := adresse_decor;
      Bounds := rectangle_limite;
      RowBytes := nbre_colonnes;
    end;
  num_lapin := 1 ;
  pic_lapin1 := lire_illustration_nr(lapin1_i);
  pic_lapin2 := lire_illustration_nr(lapin2_i);
  if pic_lapin1 <> nil then
    rect_lapin1 := pic_lapin1^^.PicFrame
  else
    SetRect(rect_lapin1 , 0, 0, 0, 0);
  larg_rect := rect_lapin1.right - rect_lapin1.left;
  haut_rect := rect_lapin1.bottom - rect_lapin1.top;
  with rect_lapin1 do
    begin
      left := (100 - larg_rect) div 2;
```

```
    right := left + larg_rect;
    top := (100 - haut_rect) div 2;
    bottom := top + haut_rect;
   end;
  if pic_lapin1 <> nil then
   rect_lapin2 := pic_lapin2^^.PicFrame
  else
   SetRect(rect_lapin2, 0, 0, 0, 0);
  larg_rect := rect_lapin2.right - rect_lapin2.left;
  haut_rect := rect_lapin2.bottom - rect_lapin2.top;
  with rect_lapin2 do
   begin
    left := (100 - larg_rect) div 2;
    right := left + larg_rect;
    top := (100 - haut_rect) div 2;
    bottom := top + haut_rect;
   end;
  with rectangle_fenetre_lapin do
   begin
    left := (rectangle_limite.left + rectangle_limite.right) div 2 - 50;
    right := (rectangle_limite.left + rectangle_limite.right) div 2 + 50;
    top := (rectangle_limite.top + rectangle_limite.bottom) div 2 - 50;
    bottom := (rectangle_limite.top + rectangle_limite.bottom) div 2 + 50;
   end;
  SetRect(rect_lapin, 2, 2, 98, 98);
  with lapin1 do
   begin
    BaseAddr := NewPtr(1152);
    RowBytes := 12;
    Bounds := rect_lapin;
   end;
  with lapin2 do
   begin
    BaseAddr := NewPtr(1152);
    RowBytes := 12;
    Bounds := rect_lapin;
   end;
  fenetre_lapin := NewWindow(nil, rectangle_fenetre_lapin, '', true, 1, nil, false, 0);
  EraseRect(rect_lapin);
  if pic_lapin1 <> nil then
   begin
    DrawPicture(pic_lapin1, rect_lapin1);
    HPurge(handle(pic_lapin1));
    ReleaseResource(Handle(pic_lapin1));
   end;
  CopyBits(port_memoire^.PortBits, lapin1, rect_lapin, rect_lapin, SrcCopy, nil);
  EraseRect(rect_lapin);
  if pic_lapin2 <> nil then
   begin
    DrawPicture(pic_lapin2, rect_lapin2);
    Hpurge(handle(pic_lapin2));
    ReleaseResource(Handle(pic_lapin2));
   end;
  CopyBits(port_memoire^.PortBits, lapin2, rect_lapin, rect_lapin, SrcCopy, nil);
  EraseRect(rectangle_limite);
  CopyBits(port_memoire^.PortBits, decor, rectangle_limite, rectangle_limite, SrcCopy, nil);
  menu_rgn := NewRgn;
  rect_menu := ScreenBits.Bounds;
  rect_menu.bottom := 20;
```

```
  RectRgn(menu_rgn, rect_menu);
  SetPort(port_sauvegarde);
end; {init_animation}

procedure retrace_ecran;

  var
   i, j: integer;
   port_sauvegarde: GrafPtr;

begin
  GetPort(port_sauvegarde);
  SetPort(port_memoire);
  CopyBits(decor, port_memoire^.PortBits, rectangle_limite, rectangle_limite, SrcCopy, nil);
  for i := 1 to nbre_pers_crees do
   begin
    j := tab_ordre_pers[i];
    if (j <= nbre_pers_crees) and (j >= 1) then
     if (tab_pers[j].pers_visible = true) then
      CopyMask(tab_pers[j].tab_attitudes[tab_pers[j].pers_attitude],
      tab_pers[j].tab_mask[tab_pers[j].pers_attitude], port_memoire^.PortBits,
      tab_pers[j].tab_attitudes[tab_pers[j].pers_attitude].Bounds,
      tab_pers[j].tab_attitudes[tab_pers[j].pers_attitude].Bounds, tab_pers[j].pers_pos);
   end;
  UnionRgn(menu_rgn, ecran_scene^.VisRgn, ecran_scene^.VisRgn);
  CopyBits(port_memoire^.PortBits, ecran_scene^.PortBits, rectangle_limite, rectangle_limite, SrcCopy,
      ecran_scene^.VisRgn);
  SetPort(ecran_scene);
  ValidRect(rectangle_limite);
  SetPort(port_sauvegarde);
end; {retrace_ecran}

procedure laisser_place_barre_menu;

  var
   i: integer;

begin
  rectangle_limite.top := ScreenBits.Bounds.top + 20;
  for i := 1 to nbre_pers_crees do
   with tab_pers[i] do
    if (pers_visible = true) and (pers_pos.top < rectangle_limite.top) then
     OffsetRect(pers_pos, 0, rectangle_limite.top - pers_pos.top);
  retrace_ecran;
end; {laisser_place_barre_menu}

procedure non_laisser_place_barre_menu;

begin
  rectangle_limite.top := ScreenBits.Bounds.top;
  retrace_ecran;
end; {non_laisser_place_barre_menu}

procedure cree_pers_complet;

  var
   port_sauvegarde: GrafPtr;
   image: PicHandle;
   rectangle: rect;
```

```
    nbre_colonnes: integer;
    taille_image: integer;
    i: integer;

begin
  GetPort(port_sauvegarde);
  SetPort(port_memoire);
  tab_pers[nr_pers_tab].pers_visible := false;
  SetRect(tab_pers[nr_pers_tab].pers_pos, 0, 0, 0, 0);
  if nr_pers_tab = nbre_pers_crees + 1 then
    nbre_pers_crees := nbre_pers_crees + 1
  else
    begin
      for i := 1 to max_attitudes do
        begin
          if (tab_pers[nr_pers_tab].tab_attitudes[i].BaseAddr) <> nil then
            DisposPtr(tab_pers[nr_pers_tab].tab_attitudes[i].BaseAddr);
          if (tab_pers[nr_pers_tab].tab_mask[i].BaseAddr) <> nil then
            DisposPtr(tab_pers[nr_pers_tab].tab_mask[i].BaseAddr);
        end;
    end;
  tab_pers[nr_pers_tab].num_pers_disque := nr_pers_disk;
  for i := 1 to max_attitudes do
    begin
      image := lire_personnage_nr_att(nr_pers_disk, i);
      if image <> nil then
        rectangle := image^^.PicFrame
      else
        SetRect(rectangle, 0, 0, 0, 0);
      OffsetRect(rectangle, -rectangle.left, -rectangle.top);
      nbre_colonnes := rectangle.right div 8;
      if (nbre_colonnes * 8) <> rectangle.right then
        nbre_colonnes := nbre_colonnes + 1;
      if (nbre_colonnes mod 2) <> 0 then
        nbre_colonnes := nbre_colonnes + 1;
      taille_image := nbre_colonnes * rectangle.bottom;
      EraseRect(rectangle);
      if image <> nil then
        begin
          DrawPicture(image, rectangle);
          Hpurge(Handle(image));
        end;
      tab_pers[nr_pers_tab].tab_attitudes[i].BaseAddr := NewPtr(taille_image);
      tab_pers[nr_pers_tab].tab_mask[i].BaseAddr := NewPtr(taille_image);
      tab_pers[nr_pers_tab].tab_attitudes[i].Bounds := rectangle;
      tab_pers[nr_pers_tab].tab_mask[i].Bounds := rectangle;
      tab_pers[nr_pers_tab].tab_attitudes[i].RowBytes := nbre_colonnes;
      tab_pers[nr_pers_tab].tab_mask[i].RowBytes := nbre_colonnes;
      CopyBits(Port_Memoire^.PortBits, tab_pers[nr_pers_tab].tab_attitudes[i], rectangle, rectangle, SrcCopy,
        nil);
      CalcMask(tab_pers[nr_pers_tab].tab_attitudes[i].BaseAddr, tab_pers[nr_pers_tab].tab_mask[i].BaseAddr,
        nbre_colonnes, nbre_colonnes, rectangle.bottom, nbre_colonnes div 2);
      change_lapin;
    end;
  tab_pers[nr_pers_tab].tete := lire_personnage_nr_att(nr_pers_disk, 19);
  SetPort(port_sauvegarde);
end; {cree_pers_complet}

procedure cree_pers_face;
```

```
 var
  port_sauvegarde : GrafPtr;
  image : PicHandle;
  rectangle : rect;
  nbre_colonnes : integer;
  taille_image : integer;
  i : integer;

begin
 GetPort(port_sauvegarde);
 SetPort(port_memoire);
 tab_pers[nr_pers_tab].pers_visible := false;
 SetRect(tab_pers[nr_pers_tab].pers_pos, 0, 0, 0, 0);
 if nr_pers_tab <= nbre_pers_crees then
  begin
   DisposPtr(tab_pers[nr_pers_tab].tab_attitudes[1].BaseAddr);
   DisposPtr(tab_pers[nr_pers_tab].tab_mask[1].BaseAddr);
  end
 else
  begin
   nbre_pers_crees := nbre_pers_crees + 1;
   for i := 2 to max_attitudes do
    begin
     tab_pers[nr_pers_tab].tab_attitudes[i].BaseAddr := nil;
     tab_pers[nr_pers_tab].tab_mask[i].BaseAddr := nil;
    end;
   tab_pers[nr_pers_tab].tete := nil;
  end;
 tab_pers[nr_pers_tab].num_pers_disque := nr_pers_disk;
 image := lire_personnage_nr_att(nr_pers_disk, 1);
 if image <> nil then
  rectangle := image^^.PicFrame
 else
  SetRect(rectangle, 0, 0, 0, 0);
 OffsetRect(rectangle, -rectangle.left, -rectangle.top);
 nbre_colonnes := rectangle.right div 8;
 if (nbre_colonnes * 8) <> rectangle.right then
  nbre_colonnes := nbre_colonnes + 1;
 if (nbre_colonnes mod 2) <> 0 then
  nbre_colonnes := nbre_colonnes + 1;
 taille_image := nbre_colonnes * rectangle.bottom;
 EraseRect(rectangle);
 if image <> nil then
  begin
   DrawPicture(image, rectangle);
   HPurge(Handle(image));
  end;
 tab_pers[nr_pers_tab].tab_attitudes[1].BaseAddr := NewPtr(taille_image);
 tab_pers[nr_pers_tab].tab_mask[1].BaseAddr := NewPtr(taille_image);
 tab_pers[nr_pers_tab].tab_attitudes[1].Bounds := rectangle;
 tab_pers[nr_pers_tab].tab_mask[1].Bounds := rectangle;
 tab_pers[nr_pers_tab].tab_attitudes[1].RowBytes := nbre_colonnes;
 tab_pers[nr_pers_tab].tab_mask[1].RowBytes := nbre_colonnes;
 CopyBits(Port_Memoire^.PortBits, tab_pers[nr_pers_tab].tab_attitudes[1], rectangle, rectangle, SrcCopy,
     nil);
 CalcMask(tab_pers[nr_pers_tab].tab_attitudes[1].BaseAddr, tab_pers[nr_pers_tab].tab_mask[1].BaseAddr,
     nbre_colonnes, nbre_colonnes, rectangle.bottom, nbre_colonnes div 2);
 SetPort(port_sauvegarde);
```

```
end; {cree_pers_face}

procedure classe_tab_ordre_pers; {procédure locale au module "animation"}

 var
  i, temp : integer;
  modif : boolean;

begin
 for i := 1 to nbre_pers_crees do
  tab_ordre_pers[i] := i;
 repeat
  modif := false;
  for i := 1 to nbre_pers_crees - 1 do
   begin
    if tab_pers[tab_ordre_pers[i]].pers_pos.bottom > tab_pers[tab_ordre_pers[i + 1]].pers_pos.bottom then
     begin
      temp := tab_ordre_pers[i];
      tab_ordre_pers[i] := tab_ordre_pers[i + 1];
      tab_ordre_pers[i + 1] := temp;
      modif := true;
     end;
   end;
 until modif = false;
end; {classe_tab_ordre_pers}

procedure detruit_personnage;

 var
  i: integer;

begin
 if nr_pers_tab <= nbre_pers_crees then
  begin
   for i := 1 to max_attitudes do
    begin
     if (tab_pers[nr_pers_tab].tab_attitudes[i].BaseAddr) <> nil then
      DisposPtr(tab_pers[nr_pers_tab].tab_attitudes[i].BaseAddr);
     if (tab_pers[nr_pers_tab].tab_mask[i].BaseAddr) <> nil then
      DisposPtr(tab_pers[nr_pers_tab].tab_mask[i].BaseAddr);
    end;
   nbre_pers_crees := nbre_pers_crees - 1;
   for i := nr_pers_tab to nbre_pers_crees do
    tab_pers[i] := tab_pers[i + 1];
   classe_tab_ordre_pers;
   retrace_ecran;
  end;
end; {detruit_personnage}

procedure place_decor;

 var
  port_sauvegarde : GrafPtr;
begin
 GetPort(port_sauvegarde);
 SetPort(port_memoire);
 EraseRect(decor.Bounds);
 if pic_decor <> nil then
```

```
    begin
      DrawPicture(pic_decor, decor.Bounds);
      HPurge(Handle(pic_decor));
      ReleaseResource(handle(pic_decor));
    end;
  CopyBits(port_memoire^.PortBits, decor, decor.Bounds, decor.Bounds, SrcCopy, nil);
  classe_tab_ordre_pers;
  retrace_ecran;
  SetPort(port_sauvegarde);
end; {place_decor}

procedure place_personnage;

  procedure calcule_rect_position (var rectangle: rect);

    var
      haut_pers, larg_pers: integer;

  begin
    rectangle := tab_pers[nr_pers].tab_attitudes[nr_attit].Bounds;
    haut_pers := rectangle.bottom - rectangle.top;
    larg_pers := rectangle.right - rectangle.left;
    if ligne <= rectangle_limite.bottom then
      if ligne - haut_pers >= rectangle_limite.top then
        begin
          rectangle.bottom := ligne;
          rectangle.top := ligne - haut_pers;
        end
      else
        begin
          rectangle.bottom := rectangle_limite.top + haut_pers;
          rectangle.top := rectangle_limite.top;
        end
    else
      begin
        rectangle.bottom := rectangle_limite.bottom;
        rectangle.top := rectangle_limite.bottom - haut_pers;
      end;
    if colonne - (larg_pers div 2) >= rectangle_limite.left then
      if colonne + (larg_pers div 2) <= rectangle_limite.right then
        begin
          rectangle.left := colonne - (larg_pers div 2);
          rectangle.right := rectangle.left + larg_pers;
        end
      else
        begin
          rectangle.left := rectangle_limite.right - larg_pers;
          rectangle.right := rectangle_limite.right;
        end
    else
      begin
        rectangle.left := rectangle_limite.left;
        rectangle.right := rectangle_limite.left + larg_pers;
      end;
  end; {calcule_rect_position}

begin {place_personnage}
  SetPort(port_memoire);
  tab_pers[nr_pers].pers_visible := true;
```

```
  tab_pers[nr_pers].pers_attitude := nr_attit;
  calcule_rect_position(tab_pers[nr_pers].pers_pos);
  classe_tab_ordre_pers;
  retrace_ecran;
end; {place_personnage}

procedure entrer_scene;

  var
    centre_ecran : integer;
    dist_horiz : integer;
    rectangle, rectangle_sauvegarde : rect;
    attitude : integer;
    intersection : boolean;

begin
  if tab_pers[nr_pers].pers_visible = false then
    begin
      tab_pers[nr_pers].pers_visible := true;
      tab_pers[nr_pers].pers_attitude := 1;
      centre_ecran := (rectangle_limite.right - rectangle_limite.left) div 2;
      rectangle := tab_pers[nr_pers].tab_attitudes[1].Bounds;
      dist_horiz := centre_ecran - (rectangle.right - rectangle.left) div 2;
      OffsetRect(rectangle, dist_horiz, rectangle_limite.Bottom);
      attitude := tab_pers[nr_pers].pers_attitude;
      rectangle_sauvegarde := tab_pers[nr_pers].tab_attitudes[attitude].Bounds;
      repeat
        OffsetRect(rectangle, 0, -taille_phase_deplacement);
        intersection := SectRect(rectangle_limite, rectangle, tab_pers[nr_pers].pers_pos);
        tab_pers[nr_pers].tab_attitudes[1].Bounds := tab_pers[nr_pers].pers_pos;
        tab_pers[nr_pers].tab_mask[1].Bounds := tab_pers[nr_pers].pers_pos;
        classe_tab_ordre_pers;
        retrace_ecran;
      until rectangle.bottom <= rectangle_limite.bottom;
      tab_pers[nr_pers].tab_attitudes[attitude].Bounds := rectangle_sauvegarde;
      tab_pers[nr_pers].tab_mask[attitude].Bounds := rectangle_sauvegarde;
    end;
end; {entree_scene}

procedure sortir_scene;

  var
    centre_ecran : integer;
    rectangle, rectangle_sauvegarde : rect;
    attitude : integer;
    intersection : boolean;

begin
  if tab_pers[nr_pers].pers_visible = true then
    begin
      centre_ecran := (rectangle_limite.right - rectangle_limite.left) div 2;
      rectangle := tab_pers[nr_pers].pers_pos;
      attitude := tab_pers[nr_pers].pers_attitude;
      rectangle_sauvegarde := tab_pers[nr_pers].tab_attitudes[attitude].Bounds;
      repeat
        OffsetRect(rectangle, 0, taille_phase_deplacement);
        intersection := SectRect(rectangle_limite, rectangle, tab_pers[nr_pers].pers_pos);
        tab_pers[nr_pers].tab_attitudes[attitude].Bounds := tab_pers[nr_pers].pers_pos;
        tab_pers[nr_pers].tab_mask[attitude].Bounds := tab_pers[nr_pers].pers_pos;
```

```
      classe_tab_ordre_pers;
      retrace_ecran;
     until rectangle.top >= rectangle_limite.bottom;
     tab_pers[nr_pers].pers_visible := false;
     SetRect(tab_pers[nr_pers].pers_pos, 0, 0, 0, 0);
     tab_pers[nr_pers].tab_attitudes[attitude].Bounds := rectangle_sauvegarde;
     tab_pers[nr_pers].tab_mask[attitude].Bounds := rectangle_sauvegarde;
    end;

end; {sortir_scene}

procedure efface_personnage;

begin
 if tab_pers[nr_pers].pers_visible = true then
  begin
   tab_pers[nr_pers].pers_visible := false;
   SetRect(tab_pers[nr_pers].pers_pos, 0, 0, 0, 0);
   classe_tab_ordre_pers;
   retrace_ecran;
  end;
end; {efface_personnage}

procedure deplace_gauche;

 var
  colonne, ligne : integer;
  colonne_initiale : integer;
  attitude_initiale : integer;

begin
 if tab_pers[nr_pers].pers_visible = true then
  begin
   colonne_initiale := (tab_pers[nr_pers].pers_pos.Right + tab_pers[nr_pers].pers_pos.Left) div 2;
   colonne := colonne_initiale;
   ligne := tab_pers[nr_pers].pers_pos.Bottom;
   attitude_initiale := tab_pers[nr_pers].pers_attitude;
   if attitude_initiale <= 10 then
    begin
     if attitude_initiale <> 4 then
      place_personnage(nr_pers, 4, ligne, colonne);
     colonne := colonne - (taille_deplacement div 4);
     place_personnage(nr_pers, 7, ligne, colonne);
     colonne := colonne - (taille_deplacement div 4);
     place_personnage(nr_pers, 4, ligne, colonne);
     colonne := colonne - (taille_deplacement div 4);
     place_personnage(nr_pers, 8, ligne, colonne);
     colonne := colonne_initiale - taille_deplacement;
     place_personnage(nr_pers, 4, ligne, colonne);
    end
   else
    repeat
     colonne := colonne - taille_phase_deplacement;
     place_personnage(nr_pers, attitude_initiale, ligne, colonne);
    until colonne <= colonne_initiale - taille_deplacement;
  end;
end; {deplace_gauche}

procedure deplace_droite;
```

```
  var
    colonne, ligne : integer;
    colonne_initiale : integer;
    attitude_initiale : integer;

begin
  if tab_pers[nr_pers].pers_visible = true then
    begin
      colonne_initiale := (tab_pers[nr_pers].pers_pos.Right + tab_pers[nr_pers].pers_pos.Left) div 2;
      colonne := colonne_initiale;
      ligne := tab_pers[nr_pers].pers_pos.Bottom;
      attitude_initiale := tab_pers[nr_pers].pers_attitude;
      if attitude_initiale <= 10 then
        begin
          if attitude_initiale <> 2 then
            place_personnage(nr_pers, 2, ligne, colonne);
          colonne := colonne + (taille_deplacement div 4);
          place_personnage(nr_pers, 5, ligne, colonne);
          colonne := colonne + (taille_deplacement div 4);
          place_personnage(nr_pers, 2, ligne, colonne);
          colonne := colonne + (taille_deplacement div 4);
          place_personnage(nr_pers, 6, ligne, colonne);
          colonne := colonne_initiale + taille_deplacement;
          place_personnage(nr_pers, 2, ligne, colonne);
        end
      else
        repeat
          colonne := colonne + taille_phase_deplacement;
          place_personnage(nr_pers, attitude_initiale, ligne, colonne);
        until colonne >= colonne_initiale + taille_deplacement;
    end;
end; {deplace_droite}

procedure deplace_haut;

  var
    ligne, colonne : integer;
    ligne_initiale : integer;
    attitude : integer;

begin
  if tab_pers[nr_pers].pers_visible = true then
    begin
      colonne := (tab_pers[nr_pers].pers_pos.Right + tab_pers[nr_pers].pers_pos.Left) div 2;
      ligne_initiale := tab_pers[nr_pers].pers_pos.Bottom;
      ligne := ligne_initiale;
      attitude := tab_pers[nr_pers].pers_attitude;
      repeat
        ligne := ligne - taille_phase_deplacement;
        place_personnage(nr_pers, attitude, ligne, colonne);
      until ligne <= ligne_initiale - taille_deplacement;
    end;
end; {deplace_haut}

procedure deplace_bas;

  var
    ligne, colonne : integer;
```

```
    ligne_initiale : integer;
    attitude : integer;

begin
  if tab_pers[nr_pers].pers_visible = true then
    begin
      colonne := (tab_pers[nr_pers].pers_pos.Right + tab_pers[nr_pers].pers_pos.Left) div 2;
      ligne_initiale := tab_pers[nr_pers].pers_pos.Bottom;
      ligne := ligne_initiale;
      attitude := tab_pers[nr_pers].pers_attitude;
      repeat
        ligne := ligne + taille_phase_deplacement;
        place_personnage(nr_pers, attitude, ligne, colonne);
      until ligne >= ligne_initiale + taille_deplacement;
    end;
end; {deplace_bas}

procedure tourne_gauche;

  var
    attitude : integer;
    ligne, colonne : integer;

begin
  if tab_pers[nr_pers].pers_visible = true then
    begin
      attitude := tab_pers[nr_pers].pers_attitude;
      case attitude of
        1 :
          attitude := 4;
        11 :
          attitude := 14;
        15 :
          attitude := 18;
        2, 3, 4, 12, 13, 14, 16, 17, 18 :
          attitude := attitude - 1;
      end;
      ligne := tab_pers[nr_pers].pers_pos.bottom;
      colonne := (tab_pers[nr_pers].pers_pos.left + tab_pers[nr_pers].pers_pos.right) div 2;
      place_personnage(nr_pers, attitude, ligne, colonne);
    end;
end; {tourne_gauche}

procedure tourne_droite;

  var
    attitude : integer;
    ligne, colonne : integer;

begin
  if tab_pers[nr_pers].pers_visible = true then
    begin
      attitude := tab_pers[nr_pers].pers_attitude;
      case attitude of
        4 :
          attitude := 1;
        14 :
          attitude := 11;
        18 :
```

```
        attitude := 15;
       1, 2, 3, 11, 12, 13, 15, 16, 17:
        attitude := attitude + 1;
      end;
      ligne := tab_pers[nr_pers].pers_pos.bottom;
      colonne := (tab_pers[nr_pers].pers_pos.left + tab_pers[nr_pers].pers_pos.right) div 2;
      place_personnage(nr_pers, attitude, ligne, colonne);
    end;

end; {tourne_droite}

procedure lever;

  var
    attitude: integer;
    ligne, colonne: integer;

begin
  if tab_pers[nr_pers].pers_visible = true then
    begin
      attitude := tab_pers[nr_pers].pers_attitude;
      case attitude of
        11, 15:
          attitude := 1;
        12, 16:
          attitude := 2;
        13, 17:
          attitude := 3;
        14, 18:
          attitude := 4;
      end;
      ligne := tab_pers[nr_pers].pers_pos.bottom;
      colonne := (tab_pers[nr_pers].pers_pos.left + tab_pers[nr_pers].pers_pos.right) div 2;
      place_personnage(nr_pers, attitude, ligne, colonne);
    end;
end; {lever}

procedure asseoir;

  var
    attitude: integer;
    ligne, colonne: integer;

begin
  if tab_pers[nr_pers].pers_visible = true then
    begin
      attitude := tab_pers[nr_pers].pers_attitude;
      case attitude of
        1, 15:
          attitude := 11;
        2, 16:
          attitude := 12;
        3, 17:
          attitude := 13;
        4, 18:
          attitude := 14;
        otherwise
      end;
      ligne := tab_pers[nr_pers].pers_pos.bottom;
```

```
    colonne := (tab_pers[nr_pers].pers_pos.left + tab_pers[nr_pers].pers_pos.right) div 2;
    place_personnage(nr_pers, attitude, ligne, colonne);
   end;
end; {asseoir}

procedure coucher;

 var
  attitude : integer;
  ligne, colonne : integer;

begin
 if tab_pers[nr_pers].pers_visible = true then
  begin
   attitude := tab_pers[nr_pers].pers_attitude;
   case attitude of
    1, 11 :
     attitude := 15;
    2, 12 :
     attitude := 16;
    3, 13 :
     attitude := 17;
    4, 14 :
     attitude := 18;
   end;
   ligne := tab_pers[nr_pers].pers_pos.bottom;
   colonne := (tab_pers[nr_pers].pers_pos.left + tab_pers[nr_pers].pers_pos.right) div 2;
   place_personnage(nr_pers, attitude, ligne, colonne);
  end;
end; {coucher}

procedure agiter;

 const
  nbre_mouvements = 2;

 var
  attitude : integer;
  ligne, colonne : integer;
  i : integer;

begin
 if tab_pers[nr_pers].pers_visible = true then
  begin
   attitude := tab_pers[nr_pers].pers_attitude;
   ligne := tab_pers[nr_pers].pers_pos.bottom;
   colonne := (tab_pers[nr_pers].pers_pos.left + tab_pers[nr_pers].pers_pos.right) div 2;
   if attitude <> 1 then
    place_personnage(nr_pers, 1, ligne, colonne);
   for i := 1 to nbre_mouvements do
    begin
     place_personnage(nr_pers, 9, ligne, colonne);
     place_personnage(nr_pers, 10, ligne, colonne);
     place_personnage(nr_pers, 1, ligne, colonne);
    end;
  end;
end; {agiter}

procedure montre_lapin;
```

```
  var
    port_sauvegarde : GrafPtr;

 begin
  GetPort(port_sauvegarde);
  SetPort(fenetre_lapin);
  BringToFront(fenetre_lapin);
  PaintOne(WindowPeek(fenetre_lapin), WindowPeek(fenetre_lapin)^.ContRgn);
  if num_lapin = 1 then
   begin
    CopyBits(lapin1 , fenetre_lapin^.PortBits, rect_lapin, rect_lapin, SrcCopy , nil);
    num_lapin := 2;
   end
  else
   begin
    CopyBits(lapin2, fenetre_lapin^.PortBits, rect_lapin, rect_lapin, SrcCopy , nil);
    num_lapin := 1 ;
   end;
  SetPort(port_sauvegarde);
 end; {montre_lapin}

 procedure change_lapin;

  var
    port_sauvegarde : GrafPtr;

 begin
  GetPort(port_sauvegarde);
  SetPort(fenetre_lapin);
  if num_lapin = 1 then
   begin
    CopyBits(lapin1 , fenetre_lapin^.PortBits, rect_lapin, rect_lapin, SrcCopy , nil);
    num_lapin := 2;
   end
  else
   begin
    CopyBits(lapin2, fenetre_lapin^.PortBits, rect_lapin, rect_lapin, SrcCopy , nil);
    num_lapin := 1 ;
   end;
  SetPort(port_sauvegarde);
 end; {change_lapin}

 procedure cache_lapin;

  var
    port_sauvegarde : GrafPtr;

 begin
  GetPort(port_sauvegarde);
  SendBehind(fenetre_lapin, nil);
  SetPort(port_sauvegarde);

 end; {cache_lapin}

end. {unit animation}
```

```
unit gestion_menus;

interface

 uses
  messages_erreurs, animation;

 procedure Init_menus;
{ Pré : /}
{ Post : la barre des menus est initialisée.}

 procedure Affiche_barre_menus;
{ Pré : /}
{ Post : la barre des menus est affichée en haut de l'écran.}

 procedure Efface_barre_menus;
{ Pré : /}
{ Post : la barre des menus est invisible sur l'écran.}

 function Selection_dans_menus (ou : point) : integer;
{ Pré : ou indique la position du clic-souris dans la barre de menus.}
{ Post : la fonction renvoie le numéro de l'item sélectionné.}

 procedure Item_accessible (nr_item : integer);
{ Pré : nr_item doit correspondre à un des numéros définis pour les items (voir module commandes).}
{ Post : l'item nr_item est accessible à l'utilisateur.}

 procedure item_non_accessible (nr_item : integer);
{ Pré : nr_item doit correspondre à un des numéros définis pour les items (voir module commandes).}
{ Post : l'item nr_item est inaccessible à l'utilisateur.}

 procedure Mettre_marque (nr_item : integer);
{ Pré : nr_item doit correspondre à un des numéros définis pour les items (voir module commandes).}
{ Post : une marque est visible à côté de l'item nr_item }

 procedure Effacer_marque (nr_item : integer);
{ Pré : nr_item doit correspondre à un des numéros définis pour les items (voir module commandes).}
{ Post : aucune marque n'est visible à côté de l'item nr_item }


implementation


 procedure ouvrir_accessoire (TheAcc : Str255);

  var
   port_gr : GrafPtr;
   MyResHandle, MyHandle : Handle;
   Size : LongInt;
   num_driver : integer;

 begin
  GetPort(port_gr);
  SetResLoad(FALSE);
  myResHandle := GetNamedResource('DRVR', theAcc);
  size := SizeResource(myResHandle);
  SetResLoad(TRUE);
  myHandle := NewHandle(size + 5120);
```

```
  if myHandle = nil then
   erreur_memoire_accessoire
  else
   begin
    DisposHandle(myHandle);
    num_driver := OpenDeskAcc(theAcc);
   end;
  SetPort(port_gr);
end;

procedure Init_menus;

 var
  menubar : handle;
  AppleMenu : MenuHandle;

begin
 CLEARMENUBAR;
 menubar := GETNEWMBAR(1);
 SETMENUBAR(menubar);
 AppleMenu := GETMHANDLE(1);
 ADDRESMENU(AppleMenu, 'DRVR');
end;



procedure Affiche_barre_menus;
begin
 LAISSER_PLACE_BARRE_MENU;
 DRAWMENUBAR
end;


procedure Efface_barre_menus;

begin
 NON_LAISSER_PLACE_BARRE_MENU;
end;


function Selection_dans_menus;

 var
  interm : longint;
  choix, menu_id, i, n_item, item_choisi : integer;
  menu_hd : menuhandle;
  marque : char;
  texte_item : STR255;

begin
 interm := MENUSELECT(ou);
 menu_id := hiword(interm);
 item_choisi := loword(interm);
 if (menu_id <> 0) then
  begin
   choix := ((menu_id * 10) + item_choisi) * 10;
   Selection_dans_menus := choix;
   HILITEMENU(0);
   if (item_choisi <= 4) and ((menu_id = 3) or (menu_id = 4)) then
```

```
      begin
       menu_hd := GETMHANDLE(menu_id);
       i := 1;
       repeat
        CHECKITEM(menu_hd, i, false);
        i := i + 1;
       until (i = 5);
       if item_choisi <= 4 then
        SETITEMMARK(menu_hd, item_choisi, chr(18))
      end;
     if (menu_id = 2) and (item_choisi = 4) then
      begin
       menu_hd := GETMHANDLE(menu_id);
       GETITEMMARK(menu_hd, 4, marque);
       if marque = chr(nomark) then
        CHECKITEM(menu_hd, 4, true)
       else
        CHECKITEM(menu_hd, 4, false);
      end;
     if (menu_id = 1) and (item_choisi > 1) then
      begin
       Selection_dans_menus := 0;
       menu_hd := GETMHANDLE(menu_id);
       GETITEM(menu_hd, item_choisi, texte_item);
       ouvrir_accessoire(texte_item);
      end;
    end
   else
    Selection_dans_menus := 0;
end;


procedure Item_accessible;

  var
   menuid: menuhandle;
   interm, nr: integer;

begin
  interm := nr_item div 100;
  menuid := GETMHANDLE(interm);
  nr := (nr_item - (interm * 100)) div 10;
  ENABLEITEM(menuid, nr);
end;


procedure item_non_accessible;
  var
   menuid: menuhandle;
   interm, nr: integer;

begin
  interm := nr_item div 100;
  menuid := GETMHANDLE(interm);
  nr := (nr_item - (interm * 100)) div 10;
  DISABLEITEM(menuid, nr);
end;
```

```
 procedure mettre_marque;

  var
   menuid : menuhandle;
   interm, nr : integer;

 begin
  interm := nr_item div 100;
  menuid := GETMHANDLE(interm);
  nr := (nr_item - (interm * 100)) div 10;
  SETITEMMARK(menuid, nr, chr(18));
 end;


 procedure effacer_marque;

  var
   menuid : menuhandle;
   interm, nr : integer;

 begin
  interm := nr_item div 100;
  menuid := GETMHANDLE(interm);
  nr := (nr_item - (interm * 100)) div 10;
  SETITEMMARK(menuid, nr, chr(nomark));
 end;
end.
```

```
unit enregistrement;

interface

 uses
  fichiers;

 type
  Tscene_enreg = record { record représentant une scène enregistrée }
     nom_enf: str255; { nom de l'utilisateur auquel appartient la scène }
     num_ref: integer;
{ numéro de la référence de la scène dans le catalogue des scènes enregistrées par nom_enf }
     reference: Tscene_ref; { référence de la scène }
     nbre_blocs: integer; { nbre de blocs de commandes que contient la scène }
     num_bloc_cour: integer; { numéro du bloc de commandes courant dans le fichier contenant la scène }
     bloc_courant: Tbloc_commandes; { bloc de commandes courant, situé en mémoire centrale }
     num_com_cour: integer; { numéro de la commande courante dans bloc_courant }
     bloc_cour_modif: boolean; { TRUE si bloc_courant a été modifié }
    end;
  Tscene_enreg_ptr = ^Tscene_enreg;
  Tfich_trace = record { record représentant une trace }
     nom_enfant: Str255; { nom de l'utilisateur }
     nom_fich: str255; { nom du fichier trace }
     tick_debut: Longint; { temps auquel la trace a commencé }
     num_ligne_cour: integer; { numéro de la ligne courante dans bloc_courant }
     bloc_courant: Tbloc_trace; { bloc de lignes courant, situé en mémoire centrale }
    end;
  Tfich_trace_ptr = ^Tfich_trace;

 function preparer_scene_enreg (nom_enfant: str255;
           num_scene_ref: integer;
           var scene_ref: Tscene_ref): Tscene_enreg_ptr;
{ Pré : 1 <= num_scene_ref <= lire_nbre_scenes (nom_enfant) + 1 }
{ Post : si num_scene_ref = lire_nbre_scenes (nom_enfant) + 1, une nouvelle scene est créée, }
{        sa référence est retournée dans scene_ref et la fonction renvoie un pointeur vers la scène. }
{        Sinon, la référence de la scène est retournée dans scene_ref et la fonction renvoie }
{        un pointeur vers la scène}

 procedure terminer_scene_enreg (scene_ptr: Tscene_enreg_ptr);
{ Pré : scene_ptr a été initialisé avec la fonction preparer_scene_enreg }
{ Post : la scène vers laquelle pointe scene_ptr est clôturée, la B.D. est mise à jour et la mémoire nettoyée }

 procedure effacer_fin_scene_enreg (scene_ptr: Tscene_enreg_ptr);
{ Pré : scene_ptr a été initialisé avec la fonction preparer_scene_enreg et la scène n'est pas clôturée }
{ Post : la fin de la scène vers laquelle pointe scene_ptr est effacée à partir de la commande courante }

 procedure ecrire_commande_scene (scene_ptr: Tscene_enreg_ptr;
           var commande: Tcommande);
{ Pré : scene_ptr a été initialisé avec la fonction preparer_scene_enreg et la scène n'est pas clôturée }
{ Post : commande est ajoutée à la fin de la scène }

 function lire_commande_scene (scene_ptr: Tscene_enreg_ptr): Tcommande;
{ Pré : scene_ptr a été initialisé avec la fonction preparer_scene_enreg et la scène n'est pas clôturée }
{ Post : la fonction renvoie la commande courante de la scène; }
{        la commande suivante devient la commande courante }

 function lire_nom_scene (scene_ptr: Tscene_enreg_ptr): str32;
{ Pré : scene_ptr a été initialisé avec la fonction preparer_scene_enreg et la scène n'est pas clôturée }
{ Post : la fonction renvoie le nom que l'utilisateur a donné a la scène vers laquelle pointe scene_ptr }
```

```
 procedure ecrire_nom_scene (scene_ptr : Tscene_enreg_ptr;
           nouv_nom : str32);
{ Pré : scene_ptr a été initialisé avec la fonction preparer_scene_enreg et la scène n'est pas clôturée }
{ Post : la scène vers laquelle pointe scene_ptrporte le nom nouv_nom }

 function preparer_fich_trace (var parametres : Tparam_enfant) : Tfich_trace_ptr;
{ Pré : / }
{ Post : un nouveau fichier trace est créé, la fonction retourne un pointeur vers la trace }

 procedure terminer_fich_trace (trace_ptr : Tfich_trace_ptr);
{ Pré : scene_ptr a été initialisé avec la fonction preparer_fich_trace }
{ Post : la trace vers laquelle pointe trace_ptr est clôturée, la B.D. est mise à jour }
{        et la mémoire est nettoyée }

 procedure trace_commande (trace_ptr : Tfich_trace_ptr;
           quand : LongInt;
           touche : integer;
           var commande : Tcommande);
{ Pré : scene_ptr a été initialisé avec la fonction preparer_fich_trace; la trace n'est pas clôturée; }
{       quand contient le moment auquel l'action à enregistrer dans la trace a été effectuée; }
{        touche contient le code de la touche qui a été éventuellement activée (0 sinon) }
{        et commande indique l'action effectuée }
{ Post : commande est enregistrée à la fin de la trace vers laquelle pointe trace_ptr}

 procedure trace_param (trace_ptr : Tfich_trace_ptr;
           quel_param : integer;
           var param : Tparam_enfant);
{ Pré : scene_ptr a été initialisé avec la fonction preparer_fich_trace; la trace n'est pas clôturée }
{ Post : le paramètre numéro quel_param de la liste de paramètres param est enregistré }
{        à la fin de la trace vers laquelle pointe trace_ptr }


implementation

 const
  ligne_oui = '----------                                    ----------';
  ligne_vide = '                                                  ';

 procedure effacer_fin_commandes (num_debut : integer;
           var bloc_com : Tbloc_commandes);

  var
   commande_vide : Tcommande;
   i : integer;

 begin
  with commande_vide do
   begin
    num_commande := -1;
    num_personnage := 0;
    attitude_pers := 0;
    ligne_pers := 0;
    colonne_pers := 0;
   end;
  for i := num_debut to nbre_com_bloc do
   bloc_com[i] := commande_vide;
end; {effacer_fin_commandes}
```

```
function preparer_scene_enreg;

 var
  scene_ptr : Tscene_enreg_ptr;

begin
 scene_ptr := Tscene_enreg_ptr(NewPtr(SizeOf(Tscene_enreg)));
 ecrire_reference_scene_enf_nr(nom_enfant, num_scene_ref, scene_ref);
 with scene_ptr^ do
  begin
   nom_enf := nom_enfant;
   num_ref := num_scene_ref;
   reference := scene_ref;
   nbre_blocs := lire_nbre_blocs_scene(reference.nom_fich);
   num_bloc_cour := 1 ;
   if nbre_blocs <> 0 then
    bloc_courant := lire_bloc_scene_nr(reference.nom_fich, 1)
   else
    effacer_fin_commandes(1 , bloc_courant);
   num_com_cour := 0;
   bloc_cour_modif := false;
  end;
 preparer_scene_enreg := scene_ptr;
end; {preparer_scene_enreg}

procedure terminer_scene_enreg;

begin
 with scene_ptr^ do
  begin
   if bloc_cour_modif = true then
    ecrire_bloc_scene_nr(reference.nom_fich, num_bloc_cour , bloc_courant);
  end;
 DisposPtr(Ptr(scene_ptr));
end; {terminer_scene_enreg}

procedure effacer_fin_scene_enreg;

begin
 with scene_ptr^ do
  begin
   if num_com_cour <> nbre_com_bloc then
    effacer_fin_commandes(num_com_cour + 1 , bloc_courant);
   if nbre_blocs > num_bloc_cour then
    effacer_a_partir_bloc_scene_nr(reference.nom_fich, num_bloc_cour + 1);
  end;
end; {effacer_fin_scene_enreg}

procedure ecrire_commande_scene;

begin
 with scene_ptr^ do
  begin
   if num_com_cour <> nbre_com_bloc then
    num_com_cour := num_com_cour + 1
   else
    begin
     if bloc_cour_modif = true then
      ecrire_bloc_scene_nr(reference.nom_fich, num_bloc_cour , bloc_courant);
```

```
        if num_bloc_cour < nbre_blocs then
         bloc_courant := lire_bloc_scene_nr(reference.nom_fich, num_bloc_cour + 1)
        else
         begin
          nbre_blocs := nbre_blocs + 1;
          effacer_fin_commandes(1, bloc_courant);
         end;
        num_com_cour := 1;
        num_bloc_cour := num_bloc_cour + 1;
       end;
     bloc_courant[num_com_cour] := commande;
     bloc_cour_modif := true;
    end;
end; {ecrire_commande_scene}

function lire_commande_scene;

 var
  commande: Tcommande;

begin
 with scene_ptr^ do
  begin
   if num_com_cour <> nbre_com_bloc then
    begin
     num_com_cour := num_com_cour + 1;
     commande := bloc_courant[num_com_cour];
    end
   else
    begin
     if bloc_cour_modif = true then
      ecrire_bloc_scene_nr(reference.nom_fich, num_bloc_cour, bloc_courant);
     if num_bloc_cour < nbre_blocs then
      begin
       num_bloc_cour := num_bloc_cour + 1;
       num_com_cour := 1;
       bloc_courant := lire_bloc_scene_nr(reference.nom_fich, num_bloc_cour);
       commande := bloc_courant[num_com_cour];
      end
     else
      with commande do
       begin
        num_commande := -1;
        num_personnage := 0;
        ligne_pers := 0;
        colonne_pers := 0;
       end;
    end;
  end;
 lire_commande_scene := commande;
end; {lire_commande_scene}

function lire_nom_scene;

begin
 lire_nom_scene := scene_ptr^.reference.nom_scene;
end; {lire_nom_scene}

procedure ecrire_nom_scene;
```

```
begin
 scene_ptr^.reference.nom_scene := nouv_nom;
 ecrire_reference_scene_enf_nr(scene_ptr^.nom_enf, scene_ptr^.num_ref, scene_ptr^.reference);
end; {ecrire_nom_scene}

procedure effacer_bloc_trace (var bloc : Tbloc_trace);

 var
  i: integer;
  texte : str255;
  adr_texte : Ptr;

begin
 texte := concat(ligne_vide, chr(13));
 adr_texte := Ptr(LongInt(@texte) + 1);
 for i := 1 to nbre_lignes_bloc_tr do
  BlockMove(adr_texte, @bloc[i], ord(texte[0]));
end; {effacer_bloc_trace}

procedure verif_bloc_trace (trace_ptr : Tfich_trace_ptr;
          nbre_lignes : integer);

begin
 if trace_ptr^.num_ligne_cour + nbre_lignes > nbre_lignes_bloc_tr then
  begin
   ecrire_bloc_trace(trace_ptr^.nom_fich, trace_ptr^.bloc_courant, nbre_lignes_bloc_tr);
   effacer_bloc_trace(trace_ptr^.bloc_courant);
   trace_ptr^.num_ligne_cour := 0;
  end;
end; {verif_bloc_trace}

procedure trace_entete (trace : Tfich_trace_ptr);

 var
  date : DateTimeRec;
  texte, date_str, heure_str, str_temp : str255;
  adr_texte : Ptr;

begin
 adr_texte := Ptr(LongInt(@texte) + 1);
 with trace^ do
  begin
   GetTime(date);
   with date do
    begin
     NumToString(day, str_temp);
     date_str := Concat(str_temp, '/');
     NumToString(month, str_temp);
     date_str := concat(date_str, str_temp);
     date_str := concat(date_str, '/');
     NumToString(year, str_temp);
     date_str := concat(date_str, str_temp);
     NumToString(hour, str_temp);
     heure_str := concat(str_temp, ':');
     NumToString(minute, str_temp);
     heure_str := concat(heure_str, str_temp);
     heure_str := concat(heure_str, ':');
     NumToString(second, str_temp);
```

```
      heure_str := concat(heure_str, str_temp);
     end;
    texte := '--- OrdiThéâtre : Fichier trace ---';
    BlockMove(adr_texte, @bloc_courant[1], ord(texte[0]));
    texte := Concat('Utilisateur : ', nom_enfant);
    BlockMove(adr_texte, @bloc_courant[2], ord(texte[0]));
    texte := Concat('Date : ', date_str);
    texte := Concat(texte, ' --- Heure : ');
    texte := Concat(texte, heure_str);
    BlockMove(adr_texte, @bloc_courant[3], ord(texte[0]));
    texte := ligne_soul;
    BlockMove(adr_texte, @bloc_courant[4], ord(texte[0]));
    num_ligne_cour := 4;
   end;
end; {trace_entete}

function num_com_to_str (num_com : integer): str255;

 var
  texte: str255;

begin
 case num_com of
  0:
   texte := '';
  1:
   texte := 'regarder une scène';
  2:
   texte := 'créer une scène';
  3:
   texte := 'gauche';
  4:
   texte := 'droite';
  5:
   texte := 'O.K.';
  6:
   texte := 'jeter';
  7:
   texte := 'stop';
  8:
   texte := 'annuler';
  9:
   texte := 'recréer';
  10:
   texte := 'musique';
  11:
   texte := 'pause';
  12:
   texte := 'décors';
  13:
   texte := 'personnages';
  14:
   texte := 'écouter';
  15:
   texte := 'changer de personnage';
  16:
   texte := 'changer ligne de commandes';
  17:
   texte := 'entrer-sortir';
```

```
18:
 texte := 'haut';
19:
 texte := 'bas';
20:
 texte := 'lever';
21:
 texte := 'asseoir';
22:
 texte := 'coucher';
23:
 texte := 'pivoter à gauche';
24:
 texte := 'pivoter à droite';
25:
 texte := 'agiter';
26:
 texte := 'caméra';
27:
 texte := 'clique sur un personnage';
28:
 texte := 'précédent';
29:
 texte := 'suivant';
31:
 texte := 'modifier';
33:
 texte := 'prendre';
34:
 texte := 'créer';
35:
 texte := 'lâcher le personnage';
100:
 texte := '<commande>-<m>';
110:
 texte := 'menu: à propos de';
210:
 texte := 'menu: lire param.';
220:
 texte := 'menu: enrég. param.';
240:
 texte := 'menu: enreg. autom.';
260:
 texte := 'menu: quitter';
310:
 texte := 'menu: mode souris';
320:
 texte := 'menu: mode clavier-flèches';
330:
 texte := 'menu: mode clavier-défilement';
350:
 texte := 'menu: vitesse de défilement';
370:
 texte := 'menu: commandes';
410:
 texte := 'menu: pas de trace';
420:
 texte := 'menu: trace sélection';
430:
```

```
    texte := 'menu: trace animation ';
   440:
    texte := 'menu: trace sélect.+ anim.';
   1001:
    texte := 'effacer personnage 1';
   1002:
    texte := 'effacer personnage 2';
   1003:
    texte := 'effacer personnage 3';
   1004:
    texte := 'effacer personnage 4';
   otherwise
    texte := '';
  end;
  num_com_to_str := texte;
end; {num_com_to_str}

procedure trace_param;
 var
  texte: str255;
  adr_texte: Ptr;

 procedure trace_mode;

  var
   mode_str: str255;

 begin
  case param.mode_util of
   1:
    mode_str := 'souris';
   2:
    mode_str := 'clavier_flèches';
   3:
    mode_str := 'clavier_défilement';
  end;
  verif_bloc_trace(trace_ptr, 1);
  trace_ptr^.num_ligne_cour := trace_ptr^.num_ligne_cour + 1;
  texte := Concat('Mode d''utilisation : ', mode_str);
  BlockMove(adr_texte, @trace_ptr^.bloc_courant[trace_ptr^.num_ligne_cour], ord(texte[0]));
 end; {trace_mode}

 procedure trace_vitesse;

  var
   vitesse_real: Real;
   vitesse_str: str255;

  procedure RealToString (num: Real;
            var str: str255);

   var
    IntTemp: LongInt;
    StrTemp: Str255;

  begin
   IntTemp := trunc(num);
   NumToString(IntTemp, str);
   IntTemp := trunc((num - IntTemp) * 100);
```

```
    if IntTemp <> 0 then
     begin
      NumToString(IntTemp, StrTemp);
      StrTemp := Concat(',', StrTemp);
      Str := Concat(Str, StrTemp);
     end;
   end;{procedure RealToString}

  begin {trace_vitesse}
   vitesse_real := param.vitesse_def / 60;
   RealToString(vitesse_real, vitesse_str);
   verif_bloc_trace(trace_ptr, 1);
   trace_ptr^.num_ligne_cour := trace_ptr^.num_ligne_cour + 1;
   texte := Concat('Temporisation du curseur (défilement) : ', vitesse_str);
   texte := concat(texte, ' sec.');
   BlockMove(adr_texte, @trace_ptr^.bloc_courant[trace_ptr^.num_ligne_cour], ord(texte[0]));
  end; {trace_vitesse}

  procedure trace_lignes_com;

   var
    i, j: integer;
    taille_texte: integer;
    fini: boolean;
    com_str: str255;

  begin
   verif_bloc_trace(trace_ptr, 1);
   trace_ptr^.num_ligne_cour := trace_ptr^.num_ligne_cour + 1;
   texte := 'Organisation des lignes de commandes : ';
   BlockMove(adr_texte, @trace_ptr^.bloc_courant[trace_ptr^.num_ligne_cour], ord(texte[0]));
   i := 1;
   fini := false;
   repeat
    j := 1;
    texte := '';
    NumToString(i, texte);
    texte := Concat('L', texte);
    texte := Concat(texte, ' : ');
    repeat
     if param.commandes[i, j] <> -1 then
      begin
       com_str := num_com_to_str(param.commandes[i, j]);
       texte := concat(texte, com_str);
       texte := Concat(texte, ' / ');
       j := j + 1
      end
     else
      fini := true;
    until (fini = true) or (j > max_com);
    taille_texte := ord(texte[0]) div nbre_car_ligne_tr;
    if ord(texte[0]) mod nbre_car_ligne_tr <> 0 then
     taille_texte := taille_texte + 1;
    verif_bloc_trace(trace_ptr, taille_texte);
    BlockMove(adr_texte, @trace_ptr^.bloc_courant[trace_ptr^.num_ligne_cour + 1], ord(texte[0]));
    trace_ptr^.num_ligne_cour := trace_ptr^.num_ligne_cour + taille_texte;
    trace_ptr^.bloc_courant[trace_ptr^.num_ligne_cour][64] := chr(13);
    if i < max_lignes then
     fini := false;
```

```
    i := i + 1;
   until (fini = true) or (i > max_lignes);
  end; {trace_lignes_com}

  procedure trace_type_tr;

   var
    type_tr_str : str255;

  begin
   case param.type_trace of
    1:
     type_tr_str := 'trace pendant la sélection';
    2:
     type_tr_str := 'trace pendant la sélection';
    3:
     type_tr_str := 'trace pendant l''animation';
    4:
     type_tr_str := 'trace pendant la sélection et l''animation';
   end;
   verif_bloc_trace(trace_ptr, 1);
   trace_ptr^.num_ligne_cour := trace_ptr^.num_ligne_cour + 1;
   texte := Concat('Type de trace : ', type_tr_str);
   BlockMove(adr_texte, @trace_ptr^.bloc_courant[trace_ptr^.num_ligne_cour], ord(texte[0]));
  end; {trace_type_tr}

  procedure trace_enreg_autom;

   var
    enreg_str : str255;

  begin
   if param.enreg_autom = true then
    enreg_str := 'oui'
   else
    enreg_str := 'non';
   verif_bloc_trace(trace_ptr, 1);
   trace_ptr^.num_ligne_cour := trace_ptr^.num_ligne_cour + 1;
   texte := Concat('Enregistrement automatique : ', enreg_str);
   BlockMove(adr_texte, @trace_ptr^.bloc_courant[trace_ptr^.num_ligne_cour], ord(texte[0]));
  end; {trace_enreg_autom}


begin {trace_param}
 adr_texte := Ptr(LongInt(@texte) + 1);
 case quel_param of
  0:
   begin
    verif_bloc_trace(trace_ptr, 1);
    trace_ptr^.num_ligne_cour := trace_ptr^.num_ligne_cour + 1;
    texte := 'Paramètres utilisés :';
    BlockMove(adr_texte, @trace_ptr^.bloc_courant[trace_ptr^.num_ligne_cour], ord(texte[0]));
    trace_mode;
    trace_vitesse;
    trace_lignes_com;
    trace_type_tr;
    trace_enreg_autom;
   end;
  2:
```

```
      trace_mode;
    3:
      trace_vitesse;
    4:
      trace_lignes_com;
    5:
      trace_type_tr;
    6:
      trace_enreg_autom;
  end;
end;{trace_param}


function preparer_fich_trace;

  var
    trace: Tfich_trace_ptr;
    texte: str255;
    adr_texte: Ptr;

begin
  trace := Tfich_trace_ptr(NewPtr(SizeOf(Tfich_trace)));
  trace^.tick_debut := TickCount;
  trace^.nom_enfant := parametres.nom_enf;
  nouv_fich_trace(parametres.nom_enf, trace^.nom_fich);
  effacer_bloc_trace(trace^.bloc_courant);
  trace^.num_ligne_cour := 0;
  trace_entete(trace);
  trace_param(trace, 0, parametres);
  verif_bloc_trace(trace, 1);
  trace^.num_ligne_cour := trace^.num_ligne_cour + 1;
  texte := ligne_soul;
  adr_texte := Ptr(LongInt(@texte) + 1);
  BlockMove(adr_texte, @trace^.bloc_courant[trace^.num_ligne_cour], ord(texte[0]));
  preparer_fich_trace := trace;
end; {preparer_fich_trace}

procedure terminer_fich_trace;

begin
  with trace_ptr^ do
    if num_ligne_cour <> 0 then
      ecrire_bloc_trace(nom_fich, bloc_courant, num_ligne_cour);
  DisposPtr(Ptr(trace_ptr));
end; {terminer_fich_trace}

procedure trace_commande;

  var
    texte: str255;
    adr_texte: Ptr;
    num_str: str255;
    point_click: point;

  function Ticks_to_str (Ticks: LongInt): str255;

    var
      min, sec, cent: integer;
      interm: LongInt;
```

```
    texte, str_temp : str255;

  begin
   min := ticks div 3600;
   interm := LongInt(min) * 3600;
   sec := (ticks - interm) div 60;
   interm := interm + (LongInt(sec) * 60);
   cent := trunc((ticks - interm) * 1.6666);
   NumToString(min, str_temp);
   texte := concat(str_temp, 'm');
   NumToString(sec, str_temp);
   texte := concat(texte, str_temp);
   texte := concat(texte, 's');
   NumToString(cent, str_temp);
   texte := concat(texte, str_temp);
   texte := concat(texte, 'c');
   ticks_to_str := texte;
  end; {ticks_to_str}

 begin {trace_commande}
  adr_texte := Ptr(LongInt(@texte) + 1);
  texte := concat(Ticks_to_str(quand - trace_ptr^.tick_debut), ' : ');
  if touche = -1 then
   begin
    texte := concat(texte, 'Clique en (');
    NumToString(commande.colonne_pers, num_str);
    texte := concat(texte, num_str);
    texte := concat(texte, ' , ');
    NumToString(commande.ligne_pers, num_str);
    texte := concat(texte, num_str);
    texte := concat(texte, ') -> ');
   end
  else
   begin
    texte := concat(texte, 'Appuie sur "');
    case touche of
     13:
      texte := concat(texte, '<retour>');
     29:
      texte := concat(texte, '<fl. à droite>');
     28:
      texte := concat(texte, '<fl. à gauche>');
     otherwise
      texte := concat(texte, chr(touche));
    end;
    texte := concat(texte, '" -> ');
   end;
  if commande.num_commande <> -1 then
   texte := concat(texte, num_com_to_str(commande.num_commande));
  verif_bloc_trace(trace_ptr, 1);
  trace_ptr^.num_ligne_cour := trace_ptr^.num_ligne_cour + 1;
  BlockMove(adr_texte, @trace_ptr^.bloc_courant[trace_ptr^.num_ligne_cour], ord(texte[0]));
 end; {trace_commande}

end. {unit enregistrement}
```

```
unit Ligne_de_commandes;

interface

 uses
  Fichiers;

 type
   ttab_com = array[1..max_com] of integer;
{tableau de maximum 8 entiers.}

   tligne_commandes = record
      nombre_com : integer;
      tab_rect : array[1..max_com] of rect;
      tab_com : ttab_com;
      tab_illustr : array[1..max_com] of pichandle;
      port_ligne : grafptr;
    end;
{record correspondant à une ligne de commandes. Il contient :}
{   le nombre de commandes sur la ligne}
{   les coordonnées des cases de commandes}
{   les numéros des commandes}
{   les illustrations des commandes}
{   un pointeur vers le port graphique dans lequel doit s'afficher la ligne de commandes}

   Ttab_voyant = array[1..max_com] of boolean;
{les éléments de ce tableau valent true lorsqu'ils correspondent à des voyants de la ligne de commandes}


 var
   Port_Graph : WindowPtr;
{Port_Graph contient un pointeur vers la fenêtre dans laquelle doit s'afficher la ligne de commandes}
   pos_curseur : integer;
{pos_curseur contient la position courante du curseur}
   ligne_cour : tligne_commandes;
{ligne_cour contient la ligne de commandes courante}
   tab_voyant : Ttab_voyant;
{tab_voyant contient le tableau indiquant s'il y a des voyants dans la ligne de commandes courante}


 procedure Init_ligne_commandes;
{ Pré : / }
{ Post : les initialisations nécessaires à l'utilisation des autres fonctions et procédures sont effectuées}

 function Defini_ligne_commandes (nbre_com : integer;
            var commandes, tab_nr_illustr : ttab_com;
            port_graphique : grafptr) : tligne_commandes;
{Pré : 1 <= nbre_com <= 8}
{       commandes est un tableau comprenant 8 numéros de commandes existants}
{       tab_nr_illustr est un tableau comprenant 8 numéros d'illustrations existants}
{       port_graphique contient un port graphique valide ou vaut nil}
{Post : la fonction renvoie un record contenant la définition de la ligne de commandes}
{       si port_graphique vaut nil, le record comprend le port graphique défini par défaut (bas de l'écran)}

 procedure Affiche_ligne_commandes (var ligne_com : tligne_commandes);
{Pré : ligne_com contient la définition valide d'une ligne de commandes}
{Post : la ligne de commandes est affichée dans le port graphique indiqué dans le record.}
```

```
 procedure Affiche_image (nr_rect: integer;
           var image: pichandle);
{Pré : nr_rect correspond à un rectangle de la ligne de commandes courante}
{Post : l'image 'image' est affichée dans le rectangle nr_rect de la ligne de commandes courante }

 procedure Place_curseur (nr_rect: integer);
{Pré : nr_rect correspond à un rectangle de la ligne de commandes courante}
{Post : le curseur est affiché dans le rectangle nr_rect de la ligne de commandes courante}

 procedure Efface_curseur;
{Pré : / }
{Post : le curseur est effacé de la ligne de commandes.}

 procedure Voyant (nr_rect: integer);
{Pré : nr_rect correspond à un rectangle de la ligne de commandes courante.}
{Post : un voyant est affiché sur le rectangle nr_rect de la ligne de commandes courante.}

 procedure Detruit_ligne_commandes (var ligne_com: tligne_commandes);
{Pré : / }
{Post : nil est affecté au port graphique de la ligne de commandes ligne_com.}


implementation

 function Defini_ligne_commandes;

  var
   large_port, large_rect, bas_port, haut_port, reste, number: integer;
   illustration: pichandle;
   i: integer;
   port_gr_ligne: grafptr;

 begin
  Defini_ligne_commandes.nombre_com := nbre_com;
  Defini_ligne_commandes.tab_com := commandes;
  if port_graphique = nil then
   port_gr_ligne := port_graph
  else
   port_gr_ligne := port_graphique;
  Defini_ligne_commandes.port_ligne := port_gr_ligne;
  large_port := (port_gr_ligne^.portrect.right) - (port_gr_ligne^.portrect.left);
  bas_port := (port_gr_ligne^.portrect.bottom);
  haut_port := (port_gr_ligne^.portrect.bottom) - 60;
  large_rect := large_port div nbre_com;
  reste := large_port - (large_rect * nbre_com);
  for i := 1 to nbre_com do
   begin
    Defini_ligne_commandes.tab_rect[i].top := haut_port;
    Defini_ligne_commandes.tab_rect[i].bottom := bas_port;
    Defini_ligne_commandes.tab_rect[i].left := large_rect * (i - 1);
    if i <> nbre_com then
     Defini_ligne_commandes.tab_rect[i].right := large_rect * i
    else
     Defini_ligne_commandes.tab_rect[i].right := (large_rect * i) + reste;
   end;
  for i := 1 to nbre_com do
   begin
    if tab_nr_illustr[i] <> 0 then
```

```
      begin
       illustration := LIRE_ILLUSTRATION_NR(tab_nr_illustr[i]);
       if illustration <> nil then
        Defini_ligne_commandes.tab_illustr[i] := illustration;
      end
     else
      Defini_ligne_commandes.tab_illustr[i] := nil;
   end
end;



procedure Calc_rectangle (var dessin: pichandle;
          var rectangle_comd, rectangle_dess: rect);

 var
  hauteur_rect, largeur_rect, haut_dessin, larg_dessin: integer;
  moitie_horiz, moitie_vertic, moitie_des_h, moitie_des_v: integer;

begin
 haut_dessin := (dessin^^.picframe.bottom) - (dessin^^.picframe.top);
 larg_dessin := (dessin^^.picframe.right) - (dessin^^.picframe.left);
 hauteur_rect := (rectangle_comd.bottom) - (rectangle_comd.top);
 largeur_rect := (rectangle_comd.right) - (rectangle_comd.left);
 moitie_horiz := (largeur_rect div 2) + rectangle_comd.left;
 moitie_vertic := (hauteur_rect div 2) + rectangle_comd.top;
 moitie_des_h := larg_dessin div 2;
 moitie_des_v := haut_dessin div 2;
 if (haut_dessin > hauteur_rect) then
  begin
   rectangle_dess.top := rectangle_comd.top;
   rectangle_dess.bottom := rectangle_comd.bottom;
   rectangle_dess.left := (moitie_horiz) - (moitie_des_h);
   rectangle_dess.right := (moitie_horiz) + (moitie_des_h);
  end
 else if (larg_dessin > largeur_rect) then
  begin
   rectangle_dess.left := rectangle_comd.left;
   rectangle_dess.right := rectangle_comd.right;
   rectangle_dess.top := (moitie_vertic) - (moitie_des_v);
   rectangle_dess.bottom := (moitie_vertic) + (moitie_des_v);
  end
 else
  begin
   rectangle_dess.left := (moitie_horiz) - (moitie_des_h);
   rectangle_dess.right := rectangle_dess.left + larg_dessin;
   rectangle_dess.top := (moitie_vertic) - (moitie_des_v);
   rectangle_dess.bottom := rectangle_dess.top + haut_dessin;
  end;
end;



procedure Affiche_ligne_commandes;

 var
  Port_Gr: grafptr;
  rectangle_com, rect_illustr: rect;
  illustration: pichandle;
```

```
    nbre_com, number : integer;
    i: integer;

begin
  GETPORT(Port_Gr);
  SETPORT(ligne_com.port_ligne);
  nbre_com := ligne_com.nombre_com;
  rectangle_com.bottom := ligne_com.port_ligne^.portrect.bottom;
  rectangle_com.top := (ligne_com.port_ligne^.portrect.bottom) - 60;
  rectangle_com.left := ligne_com.port_ligne^.portrect.left;
  rectangle_com.right := ligne_com.port_ligne^.portrect.right;
  ERASERECT(rectangle_com);
  for i := 1 to nbre_com do
    begin
      tab_voyant[i] := false;
      rectangle_com := ligne_com.tab_rect[i];
      FRAMERECT(rectangle_com);
      if (ligne_com.tab_illustr[i] <> nil) then
        begin
          illustration := ligne_com.tab_illustr[i];
          CALC_RECTANGLE(illustration, rectangle_com, rect_illustr);
          DRAWPICTURE(illustration, rect_illustr);
        end;
    end;
  VALIDRECT(ligne_com.port_ligne^.portrect);
  SETPORT(Port_Gr);
  ligne_cour := ligne_com;
  pos_curseur := 0;
end;



procedure Affiche_image;

  var
    Port_Gr : grafptr;
    rectangle_com, rect_image : rect;

begin
  if image <> nil then
    begin
      GETPORT(Port_Gr);
      SETPORT(ligne_cour.port_ligne);
      rectangle_com := ligne_cour.tab_rect[nr_rect];
      ERASERECT(rectangle_com);
      FRAMERECT(rectangle_com);
      CALC_RECTANGLE(image, rectangle_com, rect_image);
      DRAWPICTURE(image, rect_image);
      if pos_curseur = nr_rect then
        begin
          pos_curseur := 0;
          PLACE_CURSEUR(nr_rect);
        end;
      ligne_cour.tab_illustr[nr_rect] := image;
      SETPORT(Port_Gr);
    end;
end;
```

```
procedure Efface_curseur;

 var
  Port_Gr : grafptr;
  rectangle_com : rect;

begin
 GETPORT(Port_Gr);
 SETPORT(ligne_cour.port_ligne);
 PENSIZE(4 , 4);
 PENMODE(PatXOr);
 if pos_curseur <> 0 then
  begin
   rectangle_com := ligne_cour.tab_rect[pos_curseur];
   FRAMERECT(rectangle_com);
  end;
 PENSIZE(1 , 1);
 PENMODE(PatCopy);
 if pos_curseur <> 0 then
  FRAMERECT(rectangle_com);
 pos_curseur := 0;
 SETPORT(Port_Gr);
end;



procedure Place_Curseur;

 var
  Port_Gr : grafptr;
  rectangle_com : rect;

begin
 EFFACE_CURSEUR;
 GETPORT(Port_Gr);
 SETPORT(ligne_cour.port_ligne);
 PENSIZE(4 , 4);
 PENMODE(PatXOr);
 rectangle_com := ligne_cour.tab_rect[nr_rect];
 FRAMERECT(rectangle_com);
 pos_curseur := nr_rect;
 PENSIZE(1 , 1);
 PENMODE(PatCopy);
 FRAMERECT(rectangle_com);
 SETPORT(Port_Gr);
end;



procedure Voyant;

 var
  Port_Gr : grafptr;
  rectangle_com : rect;
```

```
 begin
  GETPORT(Port_Gr);
  SETPORT(ligne_cour.port_ligne);
  rectangle_com := ligne_cour.tab_rect[nr_rect];
  INVERTRECT(rectangle_com);
  FRAMERECT(rectangle_com);
  tab_voyant[nr_rect] := not tab_voyant[nr_rect];
  SETPORT(Port_Gr);
 end;



 procedure Detruit_ligne_commandes;

  var
   i: integer;
   illustr: pichandle;

 begin
  ligne_com.port_ligne := nil;
 end;



 procedure Init_ligne_commandes;


  var
   Port_Gr: grafptr;
   larg_ecran: integer;
   rect_ligne_com: rect;

 begin
  larg_ecran := (Screenbits.bounds.right) - (Screenbits.bounds.left);
  with rect_ligne_com do
   begin
    left := 0;
    right := 512;
    bottom := 342;
    top := 282;
   end;
  OffsetRect(rect_ligne_com, (ScreenBits.Bounds.Right - 512) div 2, (ScreenBits.Bounds.Bottom - 342) div 2);
  GETPORT(Port_Gr);
  port_graph := NEWWINDOW(nil, rect_ligne_com, '', true, 2, WindowPtr(-1), false, 0);
  SETPORT(Port_Gr);
  ligne_cour.port_ligne := nil;
 end;

end.
```

```
unit commandes;

interface

 uses
  Gestion_menus, Fichiers, musique, animation, enregistrement, ligne_de_commandes;

 const
{ Les constantes suivantes affectent un numéro à chaque commande se situant dans }
{ une des lignes de commandes du logiciel, ainsi qu'aux différentes options des menus. }

   fausse_com = 0; { Cette constante est utilisée pour désigner des cases se situant dans }
                   { les lignes de commandes et comprenant une icône, mais n 'étant pas}
                   { réellement des commandes. }
   regarder_scene = 1;
   creer_scene = 2;
   gauche = 3;
   droite = 4;
   ok = 5;
   jeter = 6;
   stop = 7;
   annuler = 8;
   recreer = 9;
   musique = 10;
   pause = 11;
   decors = 12;
   personnages = 13;
   ecouter = 14;
   change_pers = 15;
   change_com = 16;
   entree_sortie = 17;
   haut = 18;
   bas = 19;
   debout = 20;
   assis = 21;
   couche = 22;
   pivote_gauche = 23;
   pivote_droite = 24;
   agite = 25;
   enregistre = 26;
   clique_pers = 27;
   precedent = 28;
   suivant = 29;
   place_pers = 30;
   modification = 31;
   efface_pers = 32;
   prendre = 33;
   creer = 34;
   relache_pers = 35;
   ctrl_m = 100;
   pomme = 110;
   lireparam = 210;
   enregparam = 220;
   enr_autom = 240;
   quitter = 260;
   mode_souris = 310;
   mode_fleches = 320;
   mode_bal = 330;
   vitesse_bal = 350;
```

```
    commandes = 370;
    pas_trace = 410;
    trace_sel = 420;
    trace_anim = 430;
    trace_tout = 440;
    nbre_max_com = 46;

  var
    menuvisible, menuaccessible, calcdelai, cache_souris : boolean;
    vitesse_defil : integer;

{Lorsque menuvisible=true, la barre de menus est affichée à l'écran et lorsque menuvisible = }
{false, la barre de menus est cachée. }
{Lorsque menuaccessible=true, la barre de menus est accessible et lorsque menuaccessible = }
{false, la barre de menus est inaccessible.}
{Lorsque calcdelai=true, le prochain moment où le curseur doit se déplacer (dans le cas du }
{défilement), s'il n'y a pas de commandes validées, est calculé. Si calcdelai = false, ce temps}
{n'est pas recalculé.}
{Lorsque cache-souris=true, le curseur de la souris est invisible sinon, il apparaît à l'écran.}
{vitesse_defil  est le temps d'attente du curseur entre deux déplacements, dans le cas du défilement.}


  function Prend_commande_souris (fichier : tscene_enreg_ptr;
              trace : Tfich_trace_ptr) : tcommande;
{Pré  : fichier est un pointeur qui renseigne une scène enregistrée ou qui vaut nil.}
{       trace est un pointeur qui renseigne une trace ou qui vaut nil.}
{Post : la fonction renvoie un record de type tcommande contenant le numéro d'une commande }
{         validée au moyen de la souris. }
{         Si aucune commande n'est validée et qu'il y a une scène enregistrée (fichier<>nil), alors}
{          la fonction contient le numéro de la commande contenue dans fichier.}
{         Si une commande est validée au moyen de la souris, la fonction contient le numéro }
{          de cette commande.}
{         Si trace<> nil, le numéro de la commande est enregistré dans la trace.}

  function Prend_commande_fleches (fichier : tscene_enreg_ptr;
              trace : Tfich_trace_ptr) : tcommande;
{Pré  : fichier est un pointeur qui renseigne une scène enregistrée ou qui vaut nil.}
{       trace est un pointeur qui renseigne une trace ou qui vaut nil.}
{Post : la fonction renvoie un record de type tcommande contenant le numéro d'une commande }
{         validée dans le mode d'utilisation clavier-flèches. }
{         Si aucune commande n'est validée et qu'il y a une scène enregistrée (fichier<>nil), alors}
{          la fonction contient le numéro de la commande contenue dans fichier.}
{         Si une commande est validée dans le mode d'utilisation clavier-flèches, la fonction }
{          contient le numéro de cette commande.}
{         Si trace<> nil, le numéro de la commande est enregistré dans la trace.}

  function Prend_commande_bal (fichier : tscene_enreg_ptr;
              trace : Tfich_trace_ptr) : tcommande;
{Pré  : fichier est un pointeur qui renseigne une scène enregistrée ou qui vaut nil.}
{       trace est un pointeur qui renseigne une trace ou qui vaut nil.}
{Post : la fonction renvoie un record de type tcommande contenant le numéro d'une commande }
{         validée dans le mode d'utilisation clavier-défilement. }
{         Si aucune commande n'est validée et qu'il y a une scène enregistrée (fichier<>nil), alors}
{          la fonction contient le numéro de la commande contenue dans fichier.}
{         Si une commande est validée dans le mode d'utilisation clavier-défilement, la fonction }
{          contient le numéro de cette commande.}
{         Si trace<> nil, le numéro de la commande est enregistré dans la trace.}

  procedure Init_commandes;
```

```
{Pré : /}
{Post : les initialisations nécessaires à l'utilisation des autres fonctions sont effectuées.}


implementation

 var
  temps : longint;


 procedure traite_update_event (var event : EventRecord);

  var
   port_sauvegarde : GrafPtr;
   fenetre : WindowPtr;
   endroit : integer;
   sauve_tab_voyant : Ttab_voyant;
   sauve_pos_curseur : integer;
   i : integer;

 begin
  GetPort(port_sauvegarde);
  fenetre := WindowPtr(event.message);
  BeginUpdate(fenetre);
  if fenetre = ecran_scene then
   retrace_ecran
  else if fenetre = port_graph then
   if ligne_cour.port_ligne <> nil then
    begin
     sauve_pos_curseur := pos_curseur;
     sauve_tab_voyant := tab_voyant;
     affiche_ligne_commandes(ligne_cour);
     for i := 1 to ligne_cour.nombre_com do
      if sauve_tab_voyant[i] = true then
       voyant(i);
     place_curseur(sauve_pos_curseur);
    end;
  EndUpdate(fenetre);
  SetPort(port_sauvegarde);
 end;



 function traitesourismenu (var event : eventrecord;
           var hok : boolean) : tcommande;

  var
   comd : integer;

 begin
  if menuaccessible = true then
   begin
    comd := 0;
    comd := Selection_dans_menus(event.where);
    traitesourismenu.num_commande := comd;
    traitesourismenu.num_personnage := 0;
    traitesourismenu.ligne_pers := event.where.v;
```

```
    traitesourismenu.colonne_pers := event.where.h;
    if comd <> 0 then
     hok := true;
   end;
end;



procedure deplace_curseur (droite : boolean);

 var
  i: integer;

begin
 if droite = true then
  begin
   if pos_curseur = ligne_cour.nombre_com then
    i := 1
   else
    i := pos_curseur + 1;
   while ligne_cour.tab_com[i] = 0 do
    if i = ligne_cour.nombre_com then
     i := 1
    else
     i := i + 1;
  end
 else
  begin
   if pos_curseur = 1 then
    i := ligne_cour.nombre_com
   else
    i := pos_curseur - 1;
   while ligne_cour.tab_com[i] = 0 do
    if i = 1 then
     i := ligne_cour.nombre_com
    else
     i := i - 1;
  end;
 PLACE_CURSEUR(i)
end;



function controle_m (var event: eventrecord): boolean;

 var
  ptrtouche: ptr;
  oui: boolean;
  touchecom: integer;

begin
 if menuaccessible = true then
  begin
   touchecom := event.modifiers;
   ptrtouche := @touchecom;
   oui := BITTST(ptrtouche, 7);
   if oui = true then
```

```
      if menuvisible = true then
       begin
        EFFACE_BARRE_MENUS;
        menuvisible := false;
        if cache_souris = true then
         HideCursor;
       end
      else
       begin
        AFFICHE_BARRE_MENUS;
        menuvisible := true;
        ShowCursor;
       end;
   end;
  CONTROLE_M := oui;
end;



function traiteclavier (var event: eventrecord;
          var hok: boolean): tcommande;

 var
  touches: KeyMap;
  touche_up: boolean;
  touche, interm: integer;
  tick: LongInt;


begin
 interm := loword(event.message);
 touche := interm mod 256;
 case touche of
  29, 76, 108:
   DEPLACE_CURSEUR(true);
  28, 75, 107:
   DEPLACE_CURSEUR(false);
  13:
   begin
    voyant(pos_curseur);
    traiteclavier.num_commande := ligne_cour.tab_com[pos_curseur];
    traiteclavier.num_personnage := 0;
    traiteclavier.ligne_pers := event.where.v;
    traiteclavier.colonne_pers := event.where.h;
    hok := true;
    voyant(pos_curseur);
   end;
  77, 109:
   begin
    if CONTROLE_M(event) then
     traiteclavier.num_commande := ctrl_m
    else
     traiteclavier.num_commande := 0;
    traiteclavier.num_personnage := 0;
    traiteclavier.ligne_pers := event.where.v;
    traiteclavier.colonne_pers := event.where.h;
   end;
  otherwise
```

```
    begin
     traiteclavier.num_commande := 0;
     traiteclavier.num_personnage := 0;
     traiteclavier.ligne_pers := event.where.v;
     traiteclavier.colonne_pers := event.where.h;
    end;
  end;
end;



function Prend_commande_fleches;

  var
   event : eventrecord;
   hok, trouve : boolean;
   i : integer;
   comm : tcommande;
   quelle_fenetre : WindowPtr;

begin
  hok := false;
  i := 1;
  if pos_curseur = 0 then
   begin
    while ligne_cour.tab_com[i] = 0 do
     i := i + 1;
    PLACE_CURSEUR(i);
   end;
  repeat
   trouve := false;
   repeat
    SYSTEMTASK;
    verifier_musique;
    trouve := GETNEXTEVENT(everyevent, event);
    if (trouve = false) and (fichier <> nil) then
     begin
      comm := LIRE_COMMANDE_SCENE(fichier);
      if comm.num_commande <> (-1) then
       begin
        Prend_commande_fleches := comm;
        trouve := true;
        hok := true;
       end;
     end;
   until trouve = true;
   if hok = false then
    case event.what of
     UpdateEvt:
      begin
       traite_update_event(event);
       hok := false;
      end;
     MouseDown:
      begin
       if FINDWINDOW(event.where, quelle_fenetre) = InSysWindow then
        SYSTEMCLICK(event, quelle_fenetre);
       if (event.where.v <= 20) and (menuvisible = true) then
```

```
          comm := TRAITESOURISMENU(event, hok);
         if trace <> nil then
          trace_commande(trace, event.when, -1, comm);
        end;
       KeyDown:
        begin
         comm := TRAITECLAVIER(event, hok);
         if trace <> nil then
          trace_commande(trace, event.when, LoWord(event.message) mod 256, comm);
        end;
       otherwise
        hok := false;
      end;
  until hok = true;
  PREND_COMMANDE_FLECHES := comm;
end;



function Traite_ctrl_m (var event: eventrecord): boolean;

  var
   touche, interm: integer;

begin
  interm := loword(event.message);
  touche := interm mod 256;
  case touche of
   77, 109:
    traite_ctrl_m := CONTROLE_M(event);
   otherwise
    traite_ctrl_m := false;
  end;
end;



function attendre_m_up (num_bouton: integer): boolean;


  var
   dans_carre: boolean;
   point_souris: point;
   rectangle: rect;

begin
  dans_carre := true;
  rectangle := ligne_cour.tab_rect[num_bouton];
  voyant(num_bouton);
  repeat
   GetMouse(point_souris);
   if PTINRECT(point_souris, rectangle) <> dans_carre then
    begin
     voyant(num_bouton);
     dans_carre := not dans_carre;
    end;
  until WaitMouseUp = false;
```

```
  if dans_carre = true then
   voyant(num_bouton);
  attendre_m_up := dans_carre;
 end;


 function Traitersouris (var event: eventrecord;
          var hok: boolean): tcommande;

  var
   port_gr: grafptr;
   rectangle, rec: rect;
   point_loc: point;
   i, j: integer;
   fin: boolean;
   vertic, horiz: integer;
   tick: LongInt;
   quelle_fenetre: WindowPtr;


 begin
{clique dans accessoire de bureau}
  if FINDWINDOW(event.where, quelle_fenetre) = InSysWindow then
   SYSTEMCLICK(event, quelle_fenetre);
{clique dans menu}
  if (event.where.v <= 20) and (menuvisible = true) then
   Traitersouris := traitesourismenu(event, hok);
{clique dans barre de commandes}
  if hok = false then
   begin
    GETPORT(port_gr);
    SETPORT(ligne_cour.port_ligne);
    point_loc := event.where;
    GLOBALTOLOCAL(point_loc);
    vertic := point_loc.v;
    horiz := point_loc.h;
    rec := ligne_cour.port_ligne^.portrect;
    if (vertic > rec.bottom - 60) and (vertic < rec.bottom) and (horiz > rec.left) and (horiz < rec.right) then
     begin
      i := 0;
      fin := false;
      repeat
       i := i + 1;
       rectangle := ligne_cour.tab_rect[i];
       fin := PTINRECT(point_loc, rectangle);
      until (fin = true) or (i = ligne_cour.nombre_com);
      if ATTENDRE_M_UP(i) = true then
       begin
        traitersouris.num_commande := ligne_cour.tab_com[i];
        traitersouris.num_personnage := 0;
        traitersouris.ligne_pers := event.where.v;
        traitersouris.colonne_pers := event.where.h;
        if ligne_cour.tab_com[i] <> 0 then
         hok := true;
       end;
     end;
   end;
```

```
{ si clique dans pers ...}
   if (hok = false) then
    begin
     SETPORT(ecran_scene);
     point_loc := event.where;
     GlobalToLocal(point_loc);
     fin := false;
     i := nbre_pers_crees + 1;
     while (i > 1) and (fin = false) do
      begin
       i := i - 1;
       j := tab_ordre_pers[i];
       if tab_pers[j].pers_visible = true then
        begin
         fin := PTINRECT(point_loc, tab_pers[j].pers_pos);
        end;
      end;
     if fin = true then
      begin
       traitersouris.num_commande := clique_pers;
       traitersouris.num_personnage := j;
       traitersouris.ligne_pers := event.where.v;
       traitersouris.colonne_pers := event.where.h;
       hok := true;
      end
     else
      begin
       traitersouris.num_commande := 0;
       traitersouris.num_personnage := 0;
       traitersouris.ligne_pers := event.where.v;
       traitersouris.colonne_pers := event.where.h;
      end;
    end;
  SETPORT(port_gr);
end;



function Prend_commande_souris;

 var
  event: eventrecord;
  hok, trouve: boolean;
  comm: tcommande;

begin
 hok := false;
 trouve := false;
 repeat
  repeat
   SYSTEMTASK;
   verifier_musique;
   trouve := GETNEXTEVENT(everyevent, event);
   if (trouve = false) and (fichier <> nil) then
    begin
     comm := LIRE_COMMANDE_SCENE(fichier);
     if comm.num_commande <> (-1) then
      begin
```

```
          Prend_commande_souris := comm;
          trouve := true;
          hok := true;
         end;
       end;
    until trouve = true;
    if hok = false then
     case event.what of
      UpdateEvt:
       begin
        traite_update_event(event);
        hok := false;
       end;
      MouseDown:
       begin
        comm := TRAITERSOURIS(event, hok);
        if trace <> nil then
         trace_commande(trace, event.when, -1, comm);
       end;
      KeyDown:
       begin
        if TRAITE_CTRL_M(event) then
         comm.num_commande := ctrl_m
        else
         comm.num_commande := 0;
        comm.num_personnage := 0;
        comm.ligne_pers := event.where.v;
        comm.colonne_pers := event.where.h;
        if trace <> nil then
         trace_commande(trace, event.when, LoWord(event.message) mod 256, comm);
       end;
      otherwise
       hok := false;
     end;
  until hok = true;
  PREND_COMMANDE_SOURIS := comm;
end;



function Prend_commande_bal;

 var
  hok, trouve: boolean;
  event: eventrecord;
  touche, interm: integer;
  comm: tcommande;
  quelle_fenetre: WindowPtr;


begin
 hok := false;
 trouve := false;
 if pos_curseur = 0 then
  DEPLACE_CURSEUR(true);
 if calcdelai = true then
  temps := TickCount + vitesse_defil;
```

```
repeat
 repeat
  trouve := GETNEXTEVENT(everyevent, event);
  SYSTEMTASK;
  verifier_musique;
  if trouve = false then
   if Tickcount >= temps then
    begin
     DEPLACE_CURSEUR(true);
     temps := TickCount + vitesse_defil;
    end
   else
    begin
     if fichier <> nil then
      begin
       comm := LIRE_COMMANDE_SCENE(fichier);
       if comm.num_commande <> (-1) then
        begin
         Prend_commande_bal := comm;
         calcdelai := false;
         trouve := true;
         hok := true;
        end
       else
        begin
         calcdelai := false;
         trouve := false;
        end;
      end;
    end
  else
   calcdelai := true;
 until (trouve = true);
 if hok = false then
  case event.what of
   UpdateEvt:
    begin
     traite_update_event(event);
     hok := false;
    end;
   MouseDown:
    begin
     if FINDWINDOW(event.where, quelle_fenetre) = InSysWindow then
      SYSTEMCLICK(event, quelle_fenetre);
     if (event.where.v <= 20) and (menuvisible = true) then
      comm := TRAITESOURISMENU(event, hok);
     if trace <> nil then
      trace_commande(trace, event.when, -1, comm);
    end;
   KeyDown:
    begin
     interm := loword(event.message);
     touche := interm mod 256;
     case touche of
      13:
       begin
        voyant(pos_curseur);
        SysBeep(0);
        comm.num_commande := ligne_cour.tab_com[pos_curseur];
```

```
            comm.num_personnage := 0;
            comm.ligne_pers := event.where.v;
            comm.colonne_pers := event.where.h;
            hok := true;
            voyant(pos_curseur);
           end;
          77, 109:
           begin
            if CONTROLE_M(event) then
             comm.num_commande := ctrl_m
            else
             comm.num_commande := 0;
            comm.num_personnage := 0;
            comm.ligne_pers := event.where.v;
            comm.colonne_pers := event.where.h;
            hok := false;
           end;
          otherwise
           begin
            comm.num_commande := 0;
            comm.num_personnage := 0;
            comm.ligne_pers := event.where.v;
            comm.colonne_pers := event.where.h;
            hok := false;
           end;
         end;
         if trace <> nil then
          trace_commande(trace, event.when, touche, comm);
        end;
       otherwise
      end;
    until hok = true;
    prend_commande_bal := comm;
  end;



  procedure init_commandes;
  begin
   menuaccessible := true;
   menuvisible := false;
   vitesse_defil := 105;
   calcdelai := true;
  end;
end.
```

```
unit gestion_dialogue;

interface

 uses
  fichiers, animation, enregistrement, ligne_de_commandes, commandes;

 function fenetre_chaine_car (num_illustr_titre: integer;
          chaine_defaut: str255): str255;
{ Pré : num_illustr_titre identifie une icône du programme (voir unit fichiers) }
{       devant être utilisée comme titre de la fenêtre de dialogue et chaîne_defaut contient }
{       la chaîne de caractère devant être proposée par défaut à l'utilisateur }
{ Post : la fonction renvoie la chaîne de caractères entrée par l'utilisateur }

 function fenetre_ok_annuler (num_illustr_titre, num_illustr_dessin, mode_util: integer;
          trace: Tfich_trace_ptr): integer;
{ Pré : num_illustr_titre et num_illustr_dessin  identifient des icônes du programme (voir unit fichiers) }
{       devant être utilisées comme titre de la fenêtre de confirmation; mode_util indique le mode  }
{       d'utilisation (compris entre 1 et 3) et trace indique la trace dans laquelle il faut enregistrer }
{       les actions (nil si pas de trace) }
{ Post : la fonction retourne OK si l'utilisateur a choisi O.K., ANNULER sinon }

 function fenetre_vitesse_defilement (vitesse_initiale: integer): integer;
{ Pré : vitesse_initiale contient la vitesse de défilement par défaut à afficher dans la fenêtre de dialogue }
{ Post : la fonction retourne la vitesse de défilement choisie par le moniteur }

 function fenetre_ligne_commandes (var tab_commandes_init: Tcommandes): Tcommandes;
{ Pré : tab_commandes_init spécifie l'organisation initiale des lignes de commandes de l'animation }
{ Post : la fonction retourne l'organisation des lignes de commandes choisie par le moniteur }

 function fenetre_enregistre_parametres (nom_defaut: str255;
          var nom_choisi: str255): boolean;
{ Pré : nom_defaut contient le nom a proposer pour l'enregistrement des paramètres }
{ Post : si la fonction retourne TRUE, le moniteur a choisi d'enregistrer les paramètres courants }
{        sous le nom nom_choisi, sinon il ne désire pas les enregistrer }

 function fenetre_lire_parametres (nom_defaut: str255;
          var nom_choisi: str255): boolean;
{ Pré : nom_defaut contient le nom a proposer pour lire de nouveaux parametres }
{ Post : si la fonction retourne TRUE, le moniteur a choisi de remplacer les paramètres courants par }
{        les paramètres liés au nom nom_choisi, sinon il ne désire pas changer les paramètres courants }

 function fenetre_pas_selectionne (personnage, decor, musique: boolean;
          mode_util: integer;
          trace: Tfich_trace_ptr): integer;
{ Pré : personnage et/ou decor et/ou trace valent TRUE s'il faut indiquer à l'utilisateur qu'il a oublié de }
{       sélectionner des personnages et/ou un décor et/ou une musique; mode_util indique le mode  }
{       d'utilisation (compris entre 1 et 3) et trace indique la trace dans laquelle il faut enregistrer }
{       les actions (nil si pas de trace) }
{ Post : la fonction retourne OK si l'utilisateur décide de continuer, ANNULER s'il décide }
{        de retourner aux sélections }

 function fenetre_quitter: boolean;
{ Pré : / }
{ Post : renvoie TRUE si l'utilisateur décide de quitter le programme }

 procedure fenetre_a_propos_de (delai: LongInt);
{ Pré : delai indique la durée maximale d'affichage de la fenêtre (en 60èmes de seconde ) }
{ Post : si délai = 0, la fenêtre de pésentation du logiciel est affichée jusqu'à ce que l'utilisateur }
```

```
{       clique dans la fenêtre }
{       sinon elle est affichée jusqu'à ce que le délai indiqué par delai soit écoulé }
{       ou jusqu'à ce que l'utilisateur clique dans la fenêtre }

implementation

 const
  fen_chaine_car_id = 300;
  fen_ok_annuler_id = 310;
  fen_ligne_com_id = 320;
  fen_vit_def_id = 330;
  fen_enr_par_id = 340;
  fen_pas_select_id = 350;
  fen_lir_par_id = 360;
  fen_a_prop_id = 370;
  fen_quitter_id = 400;
  bouton_ok = 1 ;
  bouton_annuler = 2;

 var
  illustration_1 , illustration_2 : PicHandle;

 function attendre_m_up (Port_gr : GrafPtr;
           rectangle : rect) : boolean;

  var
   dans_carre : boolean;
   point_souris : point;
   port_sauvegarde : GrafPtr;

begin
 GetPort(port_sauvegarde);
 SetPort(port_gr);
 dans_carre := true;
 InvertRect(rectangle);
 repeat
  GetMouse(point_souris);
  if PTINRECT(point_souris, rectangle) <> dans_carre then
   begin
    InvertRect(rectangle);
    dans_carre := not dans_carre;
   end;
 until WaitMouseUp = false;
 if dans_carre = true then
  InvertRect(rectangle);
 SetPort(port_sauvegarde);
 attendre_m_up := dans_carre;
end;

function calc_rectangle_trace_item (rectangle_illustr, rectangle_item : rect) : rect;

 var
  delta : integer;
  rectangle : rect;

begin
 rectangle := rectangle_illustr;
 delta := (rectangle_item.bottom - rectangle_item.top) - (rectangle.bottom - rectangle.top);
 if delta <= 0 then
```

```
    begin
     rectangle.top := rectangle_item.top;
     rectangle.bottom := rectangle_item.bottom;
    end
   else
    begin
     rectangle.top := rectangle_item.top + delta div 2;
     rectangle.bottom := rectangle_item.bottom - delta div 2 - delta mod 2;
    end;
   delta := (rectangle_item.right - rectangle_item.left) - (rectangle.right - rectangle.left);
   if delta <= 0 then
    begin
     rectangle.right := rectangle_item.right;
     rectangle.left := rectangle_item.left;
    end
   else
    begin
     rectangle.right := rectangle_item.right - delta div 2 - delta mod 2;
     rectangle.left := rectangle_item.left + delta div 2;
    end;
   calc_rectangle_trace_item := rectangle;
end;

procedure Trace_item (fenetre: WindowPtr;
          item_num: integer); {procedure locale}

 var
  rectangle: rect;
  rectangle_item: rect;
  type_item: integer;
  item_handle: Handle;

begin
 GetDItem(fenetre, item_num, type_item, item_handle, rectangle_item);
 case item_num of
  1:
   if illustration_1 <> nil then
    begin
     rectangle := calc_rectangle_trace_item(illustration_1^^.PicFrame, rectangle_item);
     DrawPicture(illustration_1, rectangle);
    end;
  2:
   begin
    if illustration_2 <> nil then
     begin
      rectangle := calc_rectangle_trace_item(illustration_2^^.PicFrame, rectangle_item);
      DrawPicture(illustration_2, rectangle);
     end;
    FrameRect(rectangle_item);
   end;
 end
end; {trace_item}

function fenetre_chaine_car;

 const
  taille_dial = SizeOf(DialogRecord);

 var
```

```
    fenetre : DialogPtr;
    mem_fenetre : array [1..taille_dial] of char;
    item_num : integer;
    item_handle : Handle;
    type_item : integer;
    chaine_saisie : str255;
    rectangle_item, rectangle_item2 : rect;
    fini : boolean;

begin
  illustration_1 := lire_illustration_nr(num_illustr_titre);
  illustration_2 := lire_illustration_nr(ok_ti);
  fenetre := GetNewDialog(fen_chaine_car_id, @mem_fenetre, WindowPtr(-1));
  ShowWindow(fenetre);
  GetDItem(fenetre, 1, type_item, item_handle, rectangle_item);
  SetDItem(fenetre, 1, type_item, @trace_item, rectangle_item);
  GetDItem(fenetre, 2, type_item, item_handle, rectangle_item2);
  SetDItem(fenetre, 2, type_item, @trace_item, rectangle_item2);
  GetDItem(fenetre, 3, type_item, item_handle, rectangle_item);
  SetIText(item_handle, chaine_defaut);
  SelIText(fenetre, 3, 0, MaxInt);
  repeat
    ModalDialog(nil, item_num);
    if item_num = 2 then
      fini := attendre_m_up(fenetre, rectangle_item2)
    else
      fini := true;
  until fini = true;
  GetIText(item_handle, chaine_saisie);
  CloseDialog(fenetre);
  fenetre_chaine_car := chaine_saisie;
end; {fenetre_chaine_car}

function fenetre_ok_annuler;

  const
    taille_dial = SizeOf(DialogRecord);

  var
    fenetre : DialogPtr;
    mem_fenetre : array [1..taille_dial] of char;
    item_num : integer;
    item_handle : Handle;
    type_item : integer;
    rectangle_item : rect;
    nbre_commandes : integer;
    tab_com, tab_illustr : ttab_com;
    ligne_commandes, sauve_ligne_com : tligne_commandes;
    sauve_menu_accessible : boolean;
    commande : Tcommande;
    sauve_curseur : integer;


begin
  sauve_curseur := pos_curseur;
  sauve_ligne_com := ligne_cour;
  illustration_1 := lire_illustration_nr(num_illustr_titre);
  if illustration_1 <> nil then
    illustration_2 := lire_illustration_nr(num_illustr_dessin);
```

```
  if illustration_2 <> nil then
    fenetre := GetNewDialog(fen_ok_annuler_id, @mem_fenetre, WindowPtr(-1));
  GetDItem(fenetre, 1, type_item, item_handle, rectangle_item);
  SetDItem(fenetre, 1, type_item, @trace_item, rectangle_item);
  GetDItem(fenetre, 2, type_item, item_handle, rectangle_item);
  SetDItem(fenetre, 2, type_item, @trace_item, rectangle_item);
  ShowWindow(fenetre);
  DrawDialog(fenetre);
  nbre_commandes := 2;
  tab_com[1] := ok;
  tab_com[2] := annuler;
  tab_illustr[1] := ok_ti;
  tab_illustr[2] := annuler_ti;
  ligne_commandes := defini_ligne_commandes(nbre_commandes, tab_com, tab_illustr, fenetre);
  affiche_ligne_commandes(ligne_commandes);
  sauve_menu_accessible := MenuAccessible;
  MenuAccessible := false;
  case mode_util of
    1:
     repeat
       commande := prend_commande_souris(nil, trace);
     until (commande.num_commande = OK) or (commande.num_commande = annuler);
    2:
     commande := prend_commande_fleches(nil, trace);
    3:
     commande := prend_commande_bal(nil, trace);
  end;
  MenuAccessible := sauve_menu_accessible;
  CloseDialog(fenetre);
  ligne_cour := sauve_ligne_com;
  detruit_ligne_commandes(ligne_commandes);
  pos_curseur := sauve_curseur;
  fenetre_ok_annuler := commande.num_commande;
end; {fenetre_ok_annuler}

function fenetre_vitesse_defilement;

  const
    delai_min = 30;
    delai_max = 330;
    change_delai = 30;
    bouton_gauche = 6;
    bouton_droite = 5;
    taille_dial = SizeOf(DialogRecord);

  var
    vitesse : integer;
    vitesse_real : real;
    vitesse_str : str255;
    fenetre : DialogPtr;
    mem_fenetre : array [1..taille_dial] of char;
    nbre_commandes : integer;
    tab_com, tab_illustr : ttab_com;
    ligne_commandes, sauve_ligne_com : tligne_commandes;
    posit_curs, sauve_pos_curseur : integer;
    temps : longint;
    trouve_ev : boolean;
    evenement : EventRecord;
    fini : boolean;
```

```
  rectangle_gauche, rectangle_droite : rect;
  item_handle : Handle;
  type_item : integer;

procedure RealToString (num : Real;
          var str : str255);

  var
   IntTemp : LongInt;
   StrTemp : Str255;

begin
  IntTemp := trunc(num);
  NumToString(IntTemp, str);
  IntTemp := trunc((num - IntTemp) * 100);
  if IntTemp <> 0 then
   begin
    NumToString(IntTemp, StrTemp);
    StrTemp := Concat(',', StrTemp);
    Str := Concat(Str, StrTemp);
   end;
end;{procedure RealToString}

procedure traite_evenement;

  var
   item_actif : boolean;
   item_num : integer;

begin
  item_actif := DialogSelect(evenement, fenetre, item_num);
  if item_actif = true then
   begin
    case item_num of
     bouton_ok :
      fini := true;
     bouton_annuler :
      begin
       vitesse := vitesse_initiale;
       fini := true;
      end;
     bouton_gauche :
      if attendre_m_up(fenetre, rectangle_gauche) = true then
       begin
        if vitesse <= delai_min then
         vitesse := delai_max
        else
         vitesse := vitesse - change_delai;
        vitesse_real := vitesse / 60;
        RealToString(vitesse_real, vitesse_str);
        ParamText(vitesse_str, '', '', '');
        DrawDialog(fenetre);
        temps := TickCount + vitesse;
       end;
     bouton_droite :
      if attendre_m_up(fenetre, rectangle_droite) = true then
       begin
        if vitesse >= delai_max then
         vitesse := delai_min
```

```
          else
            vitesse := vitesse + change_delai;
          vitesse_real := vitesse / 60;
          RealToString(vitesse_real, vitesse_str);
          ParamText(vitesse_str, '', '', '');
          DrawDialog(fenetre);
          temps := TickCount + vitesse;
        end;
      end;
    end;
  end;

begin
  sauve_pos_curseur := pos_curseur;
  sauve_ligne_com := ligne_cour;
  fenetre := GetNewDialog(fen_vit_def_id, @mem_fenetre, WindowPtr(-1));
  GetDItem(fenetre, bouton_gauche, type_item, item_handle, rectangle_gauche);
  GetDItem(fenetre, bouton_droite, type_item, item_handle, rectangle_droite);
  ShowWindow(fenetre);
  vitesse_real := vitesse_initiale / 60;
  RealToString(vitesse_real, vitesse_str);
  ParamText(vitesse_str, '', '', '');
  DrawDialog(fenetre);
  nbre_commandes := 3;
  tab_com[1] := fausse_com;
  tab_com[2] := fausse_com;
  tab_com[3] := fausse_com;
  tab_illustr[1] := 0;
  tab_illustr[2] := 0;
  tab_illustr[3] := 0;
  ligne_commandes := defini_ligne_commandes(nbre_commandes, tab_com, tab_illustr, fenetre);
  affiche_ligne_commandes(ligne_commandes);
  vitesse := vitesse_initiale;
  posit_curs := 1;
  place_curseur(posit_curs);
  temps := TickCount + vitesse;
  fini := false;
  repeat
    repeat
      trouve_ev := GetNextEvent(MDownMask, evenement);
    until (trouve_ev = true) or (TickCount >= temps);
    if trouve_ev = false then
      begin
        if posit_curs = 3 then
          posit_curs := 1
        else
          posit_curs := posit_curs + 1;
        place_curseur(posit_curs);
        temps := TickCount + vitesse;
      end;
    if IsDialogEvent(evenement) then
      traite_evenement;
  until fini = true;
  CloseDialog(fenetre);
  ligne_cour := sauve_ligne_com;
  pos_curseur := sauve_pos_curseur;
  detruit_ligne_commandes(ligne_commandes);
  fenetre_vitesse_defilement := vitesse;
end; {fenetre_vitesse_defilement}
```

```
function fenetre_ligne_commandes;

 const
  bouton_ligne_suivante = 3;
  bouton_effacer = 28;
  bouton_effacer_ligne = 29;
  taille_dial = SizeOf(DialogRecord);

 type
  Titem_com = record
     numero_com : integer;
     selectionne : boolean;
     illustration : PicHandle;
     rectangle_illustr : rect;
     rectangle_item : rect;
    end;
  Ttab_item_com = array[4..25] of Titem_com;

 var
  fenetre : DialogPtr;
  mem_fenetre : array[1..taille_dial] of char;
  tab_item_com : Ttab_item_com;
  tab_commandes_fin : Tcommandes;
  ligne_courante : integer;
  evenement : EventRecord;
  item_num : integer;
  port_sauvegarde : GrafPtr;
  fini : boolean;
  tab_com : Ttab_com;
  i : integer;
  bouton_effacer_hdl : ControlHandle;

 function cherche_illustration (numero_commande : integer) : PicHandle;

  var
   num_illustr : integer;

 begin
  case numero_commande of
   change_com :
    num_illustr := change_com_i;
   entree_sortie :
    num_illustr := entree_sortie_i;
   gauche :
    num_illustr := gauche_i;
   droite :
    num_illustr := droite_i;
   haut :
    num_illustr := haut_i;
   bas :
    num_illustr := bas_i;
   debout :
    num_illustr := debout_i;
   assis :
    num_illustr := assis_i;
   couche :
    num_illustr := couche_i;
   pivote_gauche :
```

```
      num_illustr := pivote_gauche_i;
    pivote_droite:
      num_illustr := pivote_droite_i;
    agite:
      num_illustr := agite_i;
    enregistre:
      num_illustr := enregistre_i;
    musique:
      num_illustr := musique_i;
    stop:
      num_illustr := stop_i;
    otherwise
      num_illustr := 0;
  end;
  if num_illustr <> 0 then
    cherche_illustration := lire_illustration_nr(num_illustr)
  else
    cherche_illustration := nil;
end;

procedure initialise_fen_ligne_com;

  var
    i: integer;
    item_num: integer;
    item_handle: Handle;
    type_item: integer;
    rectangle_item: rect;
    ligne_courante_str: str255;

begin
  tab_commandes_fin := tab_commandes_init;
  ligne_courante := 1;
  NumToString(ligne_courante, ligne_courante_str);
  ParamText(ligne_courante_str, '', '', '');
  fenetre := GetNewDialog(fen_ligne_com_id, @mem_fenetre, WindowPtr(-1));
  ShowWindow(fenetre);
  DrawDialog(fenetre);
  GetPort(port_sauvegarde);
  SetPort(fenetre);
  GetDItem(fenetre, bouton_effacer, type_item, item_handle, rectangle_item);
  bouton_effacer_hdl := ControlHandle(item_handle);
  HiliteControl(bouton_effacer_hdl, 254);
  for i := 4 to 10 do
    tab_item_com[i].numero_com := tab_commandes_fin[1, i - 2];
  tab_item_com[11].numero_com := change_com;
  tab_item_com[12].numero_com := gauche;
  tab_item_com[13].numero_com := droite;
  tab_item_com[14].numero_com := haut;
  tab_item_com[15].numero_com := bas;
  tab_item_com[16].numero_com := entree_sortie;
  tab_item_com[17].numero_com := debout;
  tab_item_com[18].numero_com := assis;
  tab_item_com[19].numero_com := couche;
  tab_item_com[20].numero_com := agite;
  tab_item_com[21].numero_com := pivote_gauche;
  tab_item_com[22].numero_com := pivote_droite;
  tab_item_com[23].numero_com := enregistre;
  tab_item_com[24].numero_com := musique;
```

```
  tab_item_com[25].numero_com := stop;
  for i := 4 to 25 do
   with tab_item_com[i] do
    begin
     selectionne := false;
     GetDItem(fenetre, i, type_item, item_handle, rectangle_item);
     illustration := cherche_illustration(numero_com);
     if illustration <> nil then
      begin
       rectangle_illustr := calc_rectangle_trace_item(illustration^^.PicFrame, rectangle_item);
       DrawPicture(illustration, rectangle_illustr);
      end;
     FrameRect(rectangle_item);
    end;
end; {initialise}

function cherche_item_selectionne : integer;

 var
  i: integer;
  trouve : boolean;

begin
 i := 3;
 trouve := false;
 repeat
  i := i + 1;
  trouve := tab_item_com[i].selectionne;
 until (i = 25) or (trouve = true);
 if trouve = true then
  cherche_item_selectionne := i
 else
  cherche_item_selectionne := 0;
end; {cherche_item_selectionne}

procedure efface_item_selectionne;

 var
  item_selectionne : integer;
  rectangle_interieur : rect;

begin
 item_selectionne := cherche_item_selectionne;
 if (item_selectionne <> 0) and (item_selectionne < 11) then
  begin
   HiliteControl(bouton_effacer_hdl, 254);
   with tab_item_com[item_selectionne] do
    begin
     selectionne := false;
     numero_com := -1;
     illustration := nil;
     with rectangle_interieur do
      begin
       top := rectangle_item.top + 1;
       bottom := rectangle_item.bottom - 1;
       left := rectangle_item.left + 1;
       right := rectangle_item.right - 1;
      end;
     EraseRect(rectangle_interieur);
```

```
      end;
    end;
  end; {efface_item_selectionne}

  procedure clique_item_com (num_item : integer);

   var
    item_selectionne : integer;
    rectangle_interieur : rect;

  begin
   item_selectionne := cherche_item_selectionne;
   if (item_selectionne > 10) and (item_selectionne <> num_item) and (num_item < 11) then
    with tab_item_com[num_item] do
     begin
      selectionne := false;
      numero_com := tab_item_com[item_selectionne].numero_com;
      illustration := tab_item_com[item_selectionne].illustration;
      with rectangle_interieur do
       begin
        top := rectangle_item.top + 1;
        bottom := rectangle_item.bottom - 1;
        left := rectangle_item.left + 1;
        right := rectangle_item.right - 1;
       end;
      EraseRect(rectangle_interieur);
      if illustration <> nil then
       begin
        rectangle_illustr := calc_rectangle_trace_item(illustration^^.PicFrame, rectangle_item);
        DrawPicture(illustration, rectangle_illustr);
       end;
     end;
   if (item_selectionne <> 0) and (item_selectionne <> num_item) then
    with tab_item_com[item_selectionne] do
     begin
      selectionne := false;
      with rectangle_interieur do
       begin
        top := rectangle_item.top + 1;
        bottom := rectangle_item.bottom - 1;
        left := rectangle_item.left + 1;
        right := rectangle_item.right - 1;
       end;
      InvertRect(rectangle_interieur);
     end;
   if (tab_item_com[num_item].numero_com <> -1) then
    with tab_item_com[num_item] do
     begin
      selectionne := not tab_item_com[num_item].selectionne;
      if (num_item < 11) and (selectionne = true) then
       HiliteControl(bouton_effacer_hdl, 0)
      else
       HiliteControl(bouton_effacer_hdl, 254);
      with rectangle_interieur do
       begin
        top := rectangle_item.top + 1;
        bottom := rectangle_item.bottom - 1;
        left := rectangle_item.left + 1;
        right := rectangle_item.right - 1;
```

```
        end;
       InvertRect(rectangle_interieur);
      end;
  end; {clique_item_com}

  procedure ligne_com_suivante;

   var
    i: integer;
    num_ligne_str: str255;
    rectangle_interieur: rect;

  begin
   HiliteControl(bouton_effacer_hdl, 254);
   for i := 4 to 10 do
    tab_commandes_fin[ligne_courante, i - 2] := tab_item_com[i].numero_com;
   if ligne_courante = max_lignes then
    ligne_courante := 1
   else
    ligne_courante := ligne_courante + 1;
   NumToString(ligne_courante, num_ligne_str);
   ParamText(num_ligne_str, '', '', '');
   DrawDialog(fenetre);
   for i := 4 to 10 do
    with tab_item_com[i] do
     begin
      selectionne := false;
      numero_com := tab_commandes_fin[ligne_courante, i - 2];
      if numero_com <> -1 then
       illustration := cherche_illustration(numero_com)
      else
       illustration := nil;
      with rectangle_interieur do
       begin
        top := rectangle_item.top + 1;
        bottom := rectangle_item.bottom - 1;
        left := rectangle_item.left + 1;
        right := rectangle_item.right - 1;
       end;
      EraseRect(rectangle_interieur);
      if illustration <> nil then
       begin
        rectangle_illustr := calc_rectangle_trace_item(illustration^^.PicFrame, rectangle_item);
        DrawPicture(illustration, rectangle_illustr);
       end;
     end;
  end; {ligne_com_suivante}

  procedure efface_ligne_com;

   var
    i: integer;

  begin
   for i := 4 to 10 do
    begin
     tab_item_com[i].selectionne := true;
     efface_item_selectionne;
    end;
```

```
  end;

  procedure determiner_commande_1;

   var
    i, j, nbre_com : integer;

  begin
   for i := 1 to max_lignes do
    begin
     nbre_com := 0;
     for j := 2 to max_com do
      if tab_commandes_fin[i, j] <> -1 then
       nbre_com := nbre_com + 1;
     if nbre_com <> 0 then
      tab_commandes_fin[i, 1] := change_pers
     else
      tab_commandes_fin[i, 1] := -1;
    end;
  end; {determiner_commandes_1_2}

begin {fenetre_ligne_de_commande}
  initialise_fen_ligne_com;
  fini := false;
  repeat
   ModalDialog(nil, item_num);
   case item_num of
    4..25:
     clique_item_com(item_num);
    bouton_ligne_suivante:
     ligne_com_suivante;
    bouton_effacer:
     efface_item_selectionne;
    bouton_effacer_ligne:
     efface_ligne_com;
    bouton_annuler:
     begin
      tab_commandes_fin := tab_commandes_init;
      fini := true;
     end;
    bouton_ok:
     begin
      for i := 4 to 10 do
       tab_commandes_fin[ligne_courante, i - 2] := tab_item_com[i].numero_com;
      determiner_commande_1;
      fini := true;
     end;
   end;
  until fini = true;
  CloseDialog(fenetre);
  SetPort(port_sauvegarde);
  fenetre_ligne_commandes := tab_commandes_fin;
end; {fenetre_ligne_commandes}

function fenetre_enregistre_parametres;

  const
   taille_dial = SizeOf(DialogRecord);
```

```
  var
    fenetre : WindowPtr;
    mem_fenetre : array [1..taille_dial] of char;
    type_item : integer;
    item_num : integer;
    item_handle : handle;
    rectangle_item : rect;

begin
  fenetre := GetNewDialog(fen_enr_par_id, @mem_fenetre, WindowPtr(-1));
  ShowWindow(fenetre);
  GetDItem(fenetre, 3, type_item, item_handle, rectangle_item);
  SetIText(item_handle, nom_defaut);
  SelIText(fenetre, 3, 0, MaxInt);
  ModalDialog(nil, item_num);
  GetIText(item_handle, nom_choisi);
  if item_num = bouton_ok then
    fenetre_enregistre_parametres := true
  else
    fenetre_enregistre_parametres := false;
  CloseDialog(fenetre);
end; {fenetre_enregistre_parametres}

function fenetre_lire_parametres;

  const
    taille_dial = SizeOf(DialogRecord);

  var
    fenetre : WindowPtr;
    mem_fenetre : array [1..taille_dial] of char;
    type_item : integer;
    item_num : integer;
    item_handle : handle;
    rectangle_item : rect;

begin
  fenetre := GetNewDialog(fen_lir_par_id, @mem_fenetre, WindowPtr(-1));
  ShowWindow(fenetre);
  GetDItem(fenetre, 3, type_item, item_handle, rectangle_item);
  SetIText(item_handle, nom_defaut);
  SelIText(fenetre, 3, 0, MaxInt);
  ModalDialog(nil, item_num);
  GetIText(item_handle, nom_choisi);
  if item_num = bouton_ok then
    fenetre_lire_parametres := true
  else
    fenetre_lire_parametres := false;
  CloseDialog(fenetre);
end; {fenetre_lire_parametres}

function fenetre_pas_selectionne;

  const
    taille_dial = SizeOf(DialogRecord);

  var
    fenetre : DialogPtr;
    mem_fenetre : array [1..taille_dial] of char;
```

```
  item_num : integer;
  item_handle : Handle;
  type_item : integer;
  rectangle_item_1 , rectangle_item_2 , rectangle_item_3 : rect;
  nbre_commandes : integer;
  tab_com , tab_illustr : ttab_com;
  ligne_commandes , sauve_ligne_com : tligne_commandes;
  port_sauvegarde : GrafPtr;
  sauve_menu_accessible : boolean;
  commande : Tcommande;

procedure trace_illustrations;

 var
  nbre_items : integer;
  illustration_1 , illustration_2 , illustration_3 : PicHandle;
  rectangle : rect;

begin
 nbre_items := 0;
 if personnage = true then
  nbre_items := 1;
 if decor = true then
  nbre_items := nbre_items + 1;
 if musique = true then
  nbre_items := nbre_items + 1;
 case nbre_items of
  1 :
   begin
    illustration_1 := nil;
    illustration_3 := nil;
    if personnage = true then
     illustration_2 := lire_illustration_nr(pers_pas_select_i)
    else if decor = true then
     illustration_2 := lire_illustration_nr(decor_pas_select_i)
    else
     illustration_2 := lire_illustration_nr(musique_pas_select_i);
   end;
  2 :
   begin
    illustration_2 := nil;
    if personnage = true then
     begin
      illustration_1 := lire_illustration_nr(pers_pas_select_i);
      if decor = true then
       illustration_3 := lire_illustration_nr(decor_pas_select_i)
      else
       illustration_3 := lire_illustration_nr(musique_pas_select_i);
     end
    else
     begin
      illustration_1 := lire_illustration_nr(decor_pas_select_i);
      illustration_3 := lire_illustration_nr(musique_pas_select_i);
     end;
   end;
  3 :
   begin
    illustration_1 := lire_illustration_nr(pers_pas_select_i);
    illustration_2 := lire_illustration_nr(decor_pas_select_i);
```

```
        illustration_3 := lire_illustration_nr(musique_pas_select_i);
       end;
     end;
     if illustration_1 <> nil then
      begin
       rectangle := calc_rectangle_trace_item(illustration_1^^.PicFrame, rectangle_item_1);
       DrawPicture(illustration_1, rectangle);
      end;
     if illustration_2 <> nil then
      begin
       rectangle := calc_rectangle_trace_item(illustration_2^^.PicFrame, rectangle_item_2);
       DrawPicture(illustration_2, rectangle);
      end;
     if illustration_3 <> nil then
      begin
       rectangle := calc_rectangle_trace_item(illustration_3^^.PicFrame, rectangle_item_3);
       DrawPicture(illustration_3, rectangle);
      end;
   end; {trace_illustrations}

begin
  sauve_ligne_com := ligne_cour;
  fenetre := GetNewDialog(fen_pas_select_id, @mem_fenetre, WindowPtr(-1));
  GetDItem(fenetre, 1, type_item, item_handle, rectangle_item_1);
  GetDItem(fenetre, 2, type_item, item_handle, rectangle_item_2);
  GetDItem(fenetre, 3, type_item, item_handle, rectangle_item_3);
  ShowWindow(fenetre);
  DrawDialog(fenetre);
  GetPort(port_sauvegarde);
  SetPort(fenetre);
  trace_illustrations;
  nbre_commandes := 2;
  tab_com[1] := ok;
  tab_com[2] := annuler;
  tab_illustr[1] := ok_ti;
  tab_illustr[2] := annuler_ti;
  ligne_commandes := defini_ligne_commandes(nbre_commandes, tab_com, tab_illustr, fenetre);
  affiche_ligne_commandes(ligne_commandes);
  sauve_menu_accessible := MenuAccessible;
  MenuAccessible := false;
  case mode_util of
   1:
    begin
     repeat
      commande := prend_commande_souris(nil, trace);
     until (commande.num_commande = OK) or (commande.num_commande = annuler);
    end;
   2:
    commande := prend_commande_fleches(nil, trace);
   3:
    commande := prend_commande_bal(nil, trace);
  end;
  MenuAccessible := sauve_menu_accessible;
  CloseDialog(fenetre);
  ligne_cour := sauve_ligne_com;
  detruit_ligne_commandes(ligne_commandes);
  SetPort(port_sauvegarde);
  fenetre_pas_selectionne := commande.num_commande;
end; {fenetre_pas_selectionne}
```

```
function fenetre_quitter;

 var
  num_bouton : integer;

begin
 num_bouton := CautionAlert(fen_quitter_id, nil);
 fenetre_quitter := (num_bouton = bouton_ok);
end; {fenetre_quitter}

procedure fenetre_a_propos_de;

 const
  taille_dial = SizeOf(DialogRecord);

 var
  fenetre : DialogPtr;
  mem_fenetre : array [1..taille_dial] of char;
  port_sauvegarde : GrafPtr;
  type_item : integer;
  item_handle : Handle;
  rectangle_item : rect;
  evenement : EventRecord;
  fini : boolean;
  logo1 , logo2, logo3 : BitMap;
  mem_logo1 , mem_logo2, mem_logo3 : array [1..3380] of char;
  temps : LongInt;

 procedure prepare_anime_logo (rectangle : rect);

  var
   port_sauvegarde : GrafPtr;
   pic_logo : PicHandle;
   taille_image : LongInt;
   nbre_colonnes : integer;
   rect_logo : rect;
   deplac_vert, deplac_horiz : integer;

 begin
  GetPort(port_sauvegarde);
  SetPort(port_memoire);
  nbre_colonnes := (rectangle.right - rectangle.left) div 8;
  if (nbre_colonnes * 8) <> (rectangle.right - rectangle.left) then
   nbre_colonnes := nbre_colonnes + 1;
  if (nbre_colonnes mod 2) <> 0 then
   nbre_colonnes := nbre_colonnes + 1;
  taille_image := nbre_colonnes * rectangle.bottom;
  with logo1 do
   begin
    BaseAddr := @mem_logo1;
    RowBytes := nbre_colonnes;
    Bounds := rectangle;
   end;
  with logo2 do
   begin
    BaseAddr := @mem_logo2;
    RowBytes := nbre_colonnes;
    Bounds := rectangle;
```

```
    end;
   with logo3 do
    begin
     BaseAddr := @mem_logo3;
     RowBytes := nbre_colonnes;
     Bounds := rectangle;
    end;
   pic_logo := GetPicture(13101);
   HNoPurge(handle(pic_logo));
   EraseRect(rectangle);
   if pic_logo <> nil then
    begin
     rect_logo := pic_logo^^.PicFrame;
     deplac_horiz := (rectangle.right - rectangle.left) div 2 - (rect_logo.right - rect_logo.left) div 2;
     deplac_vert := (rectangle.bottom - rectangle.top) div 2 - (rect_logo.bottom - rect_logo.top) div 2;
     deplac_horiz := deplac_horiz + 5;
     deplac_vert := deplac_vert - 10;
     OffsetRect(rect_logo, deplac_horiz, deplac_vert);
     DrawPicture(pic_logo, rect_logo);
     Hpurge(Handle(pic_logo));
     ReleaseResource(Handle(pic_logo));
    end;
   CopyBits(port_memoire^.PortBits, logo1, rectangle, rectangle, SrcCopy, nil);
   pic_logo := GetPicture(13102);
   HNoPurge(handle(pic_logo));
   EraseRect(rectangle);
   if pic_logo <> nil then
    begin
     rect_logo := pic_logo^^.PicFrame;
     deplac_horiz := (rectangle.right - rectangle.left) div 2 - (rect_logo.right - rect_logo.left) div 2;
     deplac_vert := (rectangle.bottom - rectangle.top) div 2 - (rect_logo.bottom - rect_logo.top) div 2;
     deplac_horiz := deplac_horiz - 16;
     OffsetRect(rect_logo, deplac_horiz, deplac_vert);
     DrawPicture(pic_logo, rect_logo);
     Hpurge(Handle(pic_logo));
     ReleaseResource(Handle(pic_logo));
    end;
   CopyBits(port_memoire^.PortBits, logo2, rectangle, rectangle, SrcCopy, nil);
   pic_logo := GetPicture(13103);
   HNoPurge(handle(pic_logo));
   EraseRect(rectangle);
   if pic_logo <> nil then
    begin
     rect_logo := pic_logo^^.PicFrame;
     deplac_horiz := (rectangle.right - rectangle.left) div 2 - (rect_logo.right - rect_logo.left) div 2;
     deplac_vert := (rectangle.bottom - rectangle.top) div 2 - (rect_logo.bottom - rect_logo.top) div 2;
     deplac_vert := deplac_vert + 1;
     OffsetRect(rect_logo, deplac_horiz, deplac_vert);
     DrawPicture(pic_logo, rect_logo);
     Hpurge(Handle(pic_logo));
     ReleaseResource(Handle(pic_logo));
    end;
   CopyBits(port_memoire^.PortBits, logo3, rectangle, rectangle, SrcCopy, nil);
   SetPort(port_sauvegarde);
  end; {prepare_anime_logo}

  procedure anime_logo (port_gr : GrafPtr;
            rectangle : rect);
```

```
    const
      delai = 8;

    var
      port_sauvegarde : GrafPtr;
      ticks : LongInt;

  begin
    GetPort(port_sauvegarde);
    SetPort(port_gr);
    CopyBits(logo1, port_gr^.PortBits, rectangle, rectangle, SrcCopy, nil);
    delay(delai, ticks);
    CopyBits(logo2, port_gr^.PortBits, rectangle, rectangle, SrcCopy, nil);
    delay(delai, ticks);
    CopyBits(logo3, port_gr^.PortBits, rectangle, rectangle, SrcCopy, nil);
    delay(delai, ticks);
    SetPort(port_sauvegarde);
  end; {anime_logo}

begin
  GetPort(port_sauvegarde);
  fenetre := GetNewDialog(fen_a_prop_id, @mem_fenetre, WindowPtr(-1));
  GetDItem(fenetre, 1, type_item, item_handle, rectangle_item);
  DrawDialog(fenetre);
  prepare_anime_logo(rectangle_item);
  fini := false;
  temps := TickCount + delai;
  repeat
    anime_logo(fenetre, rectangle_item);
    if GetNextEvent(MDownMask, evenement) = true then
      if IsDialogEvent(evenement) then
        fini := true
      else
    else if delai <> 0 then
      fini := (TickCount >= temps);
  until fini = true;
  CloseDialog(fenetre);
  SetPort(port_sauvegarde);
end; {fenetre_a_propos_de}

end. {module gestion_dialogue}
```

```
unit ecrans;

interface

 uses
  fichiers, animation;

 var
  decor_cour, pers_cour, musique_cour, scene_cour : integer;
{ numéros du dernier décor, du dernier personnage, de la dernière musique }
{ et de la dernière scène ayant été présentés à l'utilisateur lors d'une sélection }

 procedure init_ecrans;
{ Pré : / }
{ Post : les autres procédures du module peuvent être utilisées }

 procedure affiche_ecran_fixe (ecran_numero : integer);
{ Pré : 1 <= ecran_numero <= 3 }
{ Post : ecran fixe portant le numéro ecran_numero est affiché }

 function defiler_decors (droite : boolean) : integer;
{ Pré : / }
{ Post : si droite = true, le decor suivant decor_cour est affiché à l'écran; }
{        decor_cour := decor_cour+1 et la fonction renvoie l'identifiant du décor affiché }
{        si droite = false, le decor précédant decor_cour est affiché à l'écran; }
{        decor_cour := decor_cour-1 et la fonction renvoie l'identifiant du décor affiché }

 function defiler_personnages (droite : boolean;
           numero : integer) : integer;
{ Pré : / }
{ Post : si droite = true, le personnage suivant pers_cour est affiché dans la case prévue; }
{        pers_cour := pers_cour+1 et la fonction renvoie l'identifiant du personnage affiché }
{        si droite = false, le personnage précédant pers_cour est affiché dans la case prévue; }
{        pers_cour := pers_cour-1 et la fonction renvoie l'identifiant du personnage affiché }

 function defiler_ecrans_musique (droite : boolean) : integer;
{ Pré : / }
{ Post : si droite = true, l'écran représentant la musique suivant musique_cour est affiché à l'écran; }
{        musique_cour := musique_cour+1 et la fonction renvoie l'identifiant de la musique présentée }
{        si droite = false, l'écran représentant la musique précédant musique_cour est affiché à l'écran; }
{        musique_cour := musique_cour-1 et la fonction renvoie l'identifiant de la musique }

 function defiler_scenes (droite : boolean;
           nom_enf : str255;
           var scene_ref : Tscene_ref) : integer;

implementation

 var
  nombre_decors, nombre_pers_disque, nombre_musiques : integer;

 procedure init_ecrans;

 begin
  nombre_decors := lire_nbre_decors;
  nombre_pers_disque := lire_nbre_personnages;
  nombre_musiques := lire_nbre_musiques;
  decor_cour := 1;
  pers_cour := 0;
```

```
  musique_cour := 0;
  scene_cour := 0;
end; {init_ecrans}

procedure affiche_ecran_fixe;

 var
  ecran_pic : PicHandle;

begin
 ecran_pic := lire_ecran_fixe_nr(ecran_numero);
 if ecran_pic <> nil then
  begin
   place_decor(ecran_pic);
  end;
end; {affiche_ecran_fixe}

function defiler_decors;

 var
  decor_pic : PicHandle;

begin
 if nombre_decors <> 0 then
  begin
   if droite = true then
    begin
     if decor_cour >= nombre_decors then
      decor_cour := 0
     else
      decor_cour := decor_cour + 1 ;
    end
   else
    begin
     if decor_cour <= 0 then
      decor_cour := nombre_decors
     else
      decor_cour := decor_cour - 1 ;
    end;
   decor_pic := lire_decor_nr(decor_cour);
   place_decor(decor_pic);
  end;
 defiler_decors := decor_cour;
end; {defiler_decors}

function defiler_personnages;

 var
  centre_ecran, hauteur_pers, ligne, colonne : integer;

begin
 if nombre_pers_disque > 0 then
  begin
   if droite = true then
    begin
     if pers_cour >= nombre_pers_disque then
      pers_cour := 1
     else
      pers_cour := pers_cour + 1 ;
```

```
      end
     else
      begin
       if pers_cour <= 1 then
        pers_cour := nombre_pers_disque
       else
        pers_cour := pers_cour - 1;
      end;
     cree_pers_face(numero, pers_cour);
     centre_ecran := (rectangle_limite.bottom - rectangle_limite.top) div 2;
     with tab_pers[numero].tab_attitudes[1].Bounds do
      hauteur_pers := bottom - top;
     ligne := centre_ecran + (hauteur_pers div 2);
     colonne := (rectangle_limite.right - rectangle_limite.left) div 10;
     place_personnage(numero, 1, ligne, colonne);
    end;
  defiler_personnages := pers_cour;
end; {defiler_personnages}

function defiler_ecrans_musique;

 var
  ecran_pic: PicHandle;

begin
 if nombre_musiques <> 0 then
  begin
   if droite = true then
    begin
     if musique_cour >= nombre_musiques then
      musique_cour := 1
     else
      musique_cour := musique_cour + 1;
    end
   else
    begin
     if musique_cour <= 1 then
      musique_cour := nombre_musiques
     else
      musique_cour := musique_cour - 1;
    end;
   ecran_pic := lire_ecran_musique_nr(musique_cour);
   if ecran_pic <> nil then
    begin
     place_decor(ecran_pic);
    end;
  end;
 defiler_ecrans_musique := musique_cour;
end; {defiler_ecrans_musique}

function defiler_scenes;

 const
  ligne = 250;

 var
  nbre_scenes: integer;
  decor_pic: PicHandle;
  colonne: integer;
```

```
  procedure ecrire_nom_scene (nom : str255);

   const
    ligne_nom = 50;

   var
    port_sauvegarde : GrafPtr;
    taille_nom : integer;
    colonne_nom : integer;
    num_font : integer;
    rectangle : rect;

  begin
   if nom <> '' then
    begin
     GetPort(port_sauvegarde);
     SetPort(ecran_scene);
     GetFNum('chicago', num_font);
     TextFont(num_font);
     TextSize(18);
     TextMode(SrcCopy);
     taille_nom := StringWidth(nom);
     colonne_nom := ((rectangle_limite.right + rectangle_limite.left) div 2) - (taille_nom div 2);
     SetRect(rectangle, colonne_nom - 8, ligne_nom - 24, colonne_nom + taille_nom + 9, ligne_nom + 14);
     EraseRect(rectangle);
     FrameRect(rectangle);
     InsetRect(rectangle, 3, 3);
     PenSize(2, 2);
     FrameRect(rectangle);
     PenNormal;
     MoveTo(colonne_nom, ligne_nom);
     DrawString(nom);
     SetPort(port_sauvegarde);
    end;
  end; {ecrire_nom_scene}

begin
 nbre_scenes := lire_nbre_scenes_enf(nom_enf);
 if scene_cour > nbre_scenes then
  scene_cour := nbre_scenes;
 if nbre_scenes <> 0 then
  begin
   if droite = true then
    if scene_cour >= nbre_scenes then
     scene_cour := 1
    else
     scene_cour := scene_cour + 1
   else if scene_cour <= 1 then
    scene_cour := nbre_scenes
   else
    scene_cour := scene_cour - 1;
   scene_ref := lire_reference_scene_enf_nr(nom_enf, scene_cour);
   while nbre_pers_crees <> 0 do
    detruit_personnage(1);
   if scene_ref.num_decor <> 0 then
    begin
     decor_pic := lire_decor_nr(scene_ref.num_decor);
     place_decor(decor_pic);
```

```
      end
     else
      place_decor(nil);
     if scene_ref.num_pers_1 <> 0 then
      begin
       cree_pers_face(1, scene_ref.num_pers_1);
       colonne := (rectangle_limite.right - rectangle_limite.left) div 8;
       place_personnage(1, 1, ligne, colonne);
      end;
     if scene_ref.num_pers_2 <> 0 then
      begin
       cree_pers_face(2, scene_ref.num_pers_2);
       colonne := (rectangle_limite.right - rectangle_limite.left) div 8 * 3;
       place_personnage(2, 1, ligne, colonne);
      end;
     if scene_ref.num_pers_3 <> 0 then
      begin
       cree_pers_face(3, scene_ref.num_pers_3);
       colonne := (rectangle_limite.right - rectangle_limite.left) div 8 * 5;
       place_personnage(3, 1, ligne, colonne);
      end;
     if scene_ref.num_pers_4 <> 0 then
      begin
       cree_pers_face(4, scene_ref.num_pers_4);
       colonne := (rectangle_limite.right - rectangle_limite.left) div 8 * 7;
       place_personnage(4, 1, ligne, colonne);
      end;
     ecrire_nom_scene(scene_ref.nom_scene);
     defiler_scenes := scene_cour;
    end
   else
    defiler_scenes := 0;
 end; {defiler_scenes}

end. {unit ecrans}
```

```
unit creativite;

interface

 uses
  fichiers, enregistrement, musique, animation, ligne_de_commandes, commandes, gestion_dialogue, ecrans,
      gestion_menus;

 var
  parametres : Tparam_enfant; { paramètres courants }
  trace : Tfich_trace_ptr; { fichier trace utilisé, nil si pas de trace }
  quitter_prog : boolean; { TRUE si l'utilisateur désire quitter le programme }

 procedure traiter_choix_menu (var nr_com : integer);
{ Pré : nr_com représente un choix dans les menus déroulants }
{ Post : le choix effectué dans les menus déroulants par le moniteur est appliqué }

 procedure creer_nouv_scene (scene_ptr : Tscene_enreg_ptr;
           var reference_scene : Tscene_ref);
{ Pré : scene_ptr pointe vers une structure de scène enregistrée et reference_scene contient }
{       la référence de cette scène }
{ Post : l'utilisateur a joué une scène, la B.D. a été mise à jour s'il a décidé d'enregistrer }

 procedure choisir_personnages;
{ Pré : / }
{ Post : le tableau tab_pers (voir unit animation) contient les personnages choisis par l'utilisateur }

 function Choisir_decor : integer;
{ Pré : / }
{ Post : la fonction renvoie le numéro du décor choisi par l'utilisateur }

 function choisir_musique : integer;
{ Pré : / }
{ Post : la fonction renvoie le numéro de la musique choisie par l'utilisateur }

 function Choisir_scene : Tscene_enreg_ptr;
{ Pré : / }
{ Post : la fonction renvoie un pointeur vers la scène choisie par l'utilisateur }
{        ou nil s'il n'a pas choisi de scène }

 function Visualiser_scene (scene_ptr : Tscene_enreg_ptr): integer;
{ Pré : scene_ptr pointe vers une scène enregistrée }
{ Post : l'utilisateur a visualisé la scène, la fonction retourne le numéro de la commande par laquelle }
{        l'utilisateur a mis fin à la visualisation (STOP ou RECREER) }


implementation

 const
  ligne = 250;

 type
  tab4 = array[1..4] of integer;


 var
  pas3comd : boolean;
  cases : array[1..5] of integer;
  place_tab : integer;
```

```
procedure appliquer_parametres;

begin
 vitesse_defil := parametres.vitesse_def;
 if parametres.enreg_autom = true then
  mettre_marque(enr_autom)
 else
  effacer_marque(enr_autom);
 case parametres.mode_util of
  1:
   begin
    mettre_marque(mode_souris);
    effacer_marque(mode_bal);
    effacer_marque(mode_fleches);
    efface_curseur;
    cache_souris := false;
   end;
  2:
   begin
    mettre_marque(mode_fleches);
    effacer_marque(mode_bal);
    effacer_marque(mode_souris);
    cache_souris := true;
   end;
  3:
   begin
    mettre_marque(mode_bal);
    effacer_marque(mode_souris);
    effacer_marque(mode_fleches);
    cache_souris := true;
   end;
 end;
 case parametres.type_trace of
  1:
   begin
    mettre_marque(pas_trace);
    effacer_marque(trace_sel);
    effacer_marque(trace_anim);
    effacer_marque(trace_tout);
    trace := nil;
   end;
  2:
   begin
    mettre_marque(trace_sel);
    effacer_marque(pas_trace);
    effacer_marque(trace_anim);
    effacer_marque(trace_tout);
    trace := nil;
   end;
  3:
   begin
    mettre_marque(trace_anim);
    effacer_marque(trace_sel);
    effacer_marque(pas_trace);
    effacer_marque(trace_tout);
    trace := nil;
   end;
  4:
```

```
    begin
     mettre_marque(trace_tout);
     effacer_marque(trace_sel);
     effacer_marque(trace_anim);
     effacer_marque(pas_trace);
     trace := preparer_fich_trace(parametres);
    end;
  end;
end;

procedure traiter_choix_menu;

 var
  nouv_com : Tcommandes;
  nouv_vitesse : integer;
  nouv_nom : str255;
  enregistrer, lire : boolean;

begin
 case nr_com of
  pomme :
   begin
    fenetre_a_propos_de(0);
   end;
  quitter :
   quitter_prog := fenetre_quitter;
  enr_autom :
   begin
    parametres.enreg_autom := not parametres.enreg_autom;
    if trace <> nil then
     trace_param(trace, 6, parametres);
   end;
  commandes :
   begin
    nouv_com := fenetre_ligne_commandes(parametres.commandes);
    parametres.commandes := nouv_com;
    if trace <> nil then
     trace_param(trace, 4, parametres);
   end;
  mode_souris :
   begin
    if (parametres.mode_util = 2) or (parametres.mode_util = 3) then
     efface_curseur;
    cache_souris := false;
    parametres.mode_util := 1;
    if trace <> nil then
     trace_param(trace, 2, parametres);
   end;
  mode_fleches :
   begin
    cache_souris := true;
    parametres.mode_util := 2;
    if trace <> nil then
     trace_param(trace, 2, parametres);
   end;
  mode_bal :
   begin
    cache_souris := true;
    parametres.mode_util := 3;
```

```
    if trace <> nil then
     trace_param(trace, 2, parametres);
   end;
  vitesse_bal:
   begin
    nouv_vitesse := fenetre_vitesse_defilement(parametres.vitesse_def);
    parametres.vitesse_def := nouv_vitesse;
    vitesse_defil := nouv_vitesse;
    if trace <> nil then
     trace_param(trace, 3, parametres);
   end;
  pas_trace:
   begin
    parametres.type_trace := 1;
    if trace <> nil then
     trace_param(trace, 5, parametres);
   end;
  trace_sel:
   begin
    parametres.type_trace := 2;
    if trace <> nil then
     trace_param(trace, 5, parametres);
   end;
  trace_anim:
   begin
    parametres.type_trace := 3;
    if trace <> nil then
     trace_param(trace, 5, parametres);
   end;
  trace_tout:
   begin
    parametres.type_trace := 4;
    if trace <> nil then
     trace_param(trace, 5, parametres);
   end;
  lireparam:
   begin
    lire := fenetre_lire_parametres(parametres.nom_enf, nouv_nom);
    if lire = true then
     begin
      parametres := lire_parametres_enfant(nouv_nom);
      if trace <> nil then
       begin
        terminer_fich_trace(trace);
        trace := nil;
       end;
      appliquer_parametres;
     end
    else
     nr_com := 0;
   end;
  enregparam:
   begin
    enregistrer := fenetre_enregistre_parametres(parametres.nom_enf, nouv_nom);
    if enregistrer = true then
     begin
      parametres.nom_enf := nouv_nom;
      ecrire_parametres_enfant(parametres);
     end
```

```
      else
        nr_com := 0;
      end;
    end;
  end; {traiter_choix_menu}

  procedure creer_nouv_scene;

    var
      num_scene : integer;
      tab_lignes_commandes : array [1..max_lignes] of Tligne_commandes;
      nbre_lignes : integer;
      enregistrer : boolean;
      a_enregistre : boolean;
      stop_creer : boolean;
      commande : Tcommande;
      num_com : integer;
      ligne_courante : integer;
      pers_courant : integer;
      i : integer;
      point_clic : point;
      posit_cur : integer;
      commandes_enregistrer : array [1..max_lignes] of integer;
      commandes_musique : array [1..max_lignes] of integer;
      nouv_nom_scene : str32;
      trace_2 : Tfich_trace_ptr;

    function cherche_illustration (numero_com : integer) : integer;

    begin
      case numero_com of
        change_com :
          cherche_illustration := change_com_i;
        change_pers :
          cherche_illustration := 0;
        entree_sortie :
          cherche_illustration := entree_sortie_i;
        gauche :
          cherche_illustration := gauche_i;
        droite :
          cherche_illustration := droite_i;
        haut :
          cherche_illustration := haut_i;
        bas :
          cherche_illustration := bas_i;
        debout :
          cherche_illustration := debout_i;
        assis :
          cherche_illustration := assis_i;
        couche :
          cherche_illustration := couche_i;
        pivote_gauche :
          cherche_illustration := pivote_gauche_i;
        pivote_droite :
          cherche_illustration := pivote_droite_i;
        agite :
          cherche_illustration := agite_i;
        enregistre :
          cherche_illustration := enregistre_i;
```

```
    musique:
      cherche_illustration := musique_i;
    stop:
      cherche_illustration := stop_i;
    otherwise
      cherche_illustration := 0;
  end;
end; {cherche_illustration}

procedure remplir_tableau_lignes;

  var
    tab_com_ligne, tab_illustr: Ttab_com;
    nbre_com, com, i, j: integer;

begin
  if nbre_pers_crees <> 0 then
    begin
      nbre_lignes := 0;
      for i := 1 to max_lignes do
        begin
          nbre_com := 0;
          commandes_enregistrer[i] := 0;
          commandes_musique[i] := 0;
          for j := 1 to max_com do
            begin
              com := parametres.commandes[i, j];
              if com <> -1 then
                begin
                  nbre_com := nbre_com + 1;
                  tab_com_ligne[nbre_com] := com;
                  tab_illustr[nbre_com] := cherche_illustration(com);
                  if com = enregistre then
                    commandes_enregistrer[i] := nbre_com
                  else if com = musique then
                    commandes_musique[i] := nbre_com;
                end;
            end;
          if nbre_com <> 0 then
            begin
              nbre_lignes := nbre_lignes + 1;
              tab_lignes_commandes[nbre_lignes] := defini_ligne_commandes(nbre_com, tab_com_ligne,
      tab_illustr, nil);
            end;
        end;
      if nbre_lignes = 0 then
        begin
          nbre_lignes := 1;
          tab_com_ligne[1] := fausse_com;
          tab_illustr[1] := 0;
          tab_com_ligne[2] := stop;
          tab_illustr[2] := stop_i;
          tab_lignes_commandes[1] := defini_ligne_commandes(2, tab_com_ligne, tab_illustr, nil);
        end;
    end
  else
    begin
      nbre_lignes := 1;
      tab_com_ligne[1] := musique;
```

```
      tab_illustr[1] := musique_i;
      tab_com_ligne[2] := stop;
      tab_illustr[2] := stop_i;
      tab_lignes_commandes[1] := defini_ligne_commandes(2, tab_com_ligne, tab_illustr, nil);
      commandes_musique[1] := 1;
    end;
  end; {remplir_tableau_lignes}

  procedure enregistrer_positions;

   var
    i: integer;

  begin
   for i := 1 to nbre_pers_crees do
    if tab_pers[i].pers_visible = false then
     begin
      commande.num_commande := efface_pers;
      commande.num_personnage := i;
      ecrire_commande_scene(scene_ptr, commande);
     end;
   for i := 1 to nbre_pers_crees do
    if tab_pers[i].pers_visible = true then
     begin
      commande.num_commande := place_pers;
      commande.num_personnage := i;
      commande.attitude_pers := tab_pers[i].pers_attitude;
      commande.ligne_pers := tab_pers[i].pers_pos.bottom;
      commande.colonne_pers := (tab_pers[i].pers_pos.left + tab_pers[i].pers_pos.right) div 2;
      ecrire_commande_scene(scene_ptr, commande);
     end;
  end; {enregistrer_positions}

  procedure suivre_souris (nr_pers: integer;
            point_clic: point);
   const
    contrainte_horiz = 15;
    contrainte_vertic = 3;

   var
    point_1, point_2: point;
    deplac_horiz, deplac_vertic: integer;
    ligne, colonne, colonne_temp: integer;
    attitude: integer;
    evenement: EventRecord;
    commande: Tcommande;

  begin
   HideCursor;
   point_1 := point_clic;
   commande.num_commande := place_pers;
   commande.num_personnage := nr_pers;
   repeat
    SYSTEMTASK;
    verifier_musique;
    with tab_pers[nr_pers].pers_pos do
     begin
      ligne := bottom;
      colonne := ((right + left) div 2);
```

```
 end;
GetMouse(point_2);
LocalToGlobal(point_2);
deplac_vertic := point_2.v - point_1.v;
deplac_horiz := point_2.h - point_1.h;
attitude := tab_pers[nr_pers].pers_attitude;
if abs(deplac_horiz) >= contrainte_horiz then
 begin
  if attitude < 11 then
   if deplac_horiz > 0 then
    begin
     colonne_temp := colonne;
     colonne := colonne + (deplac_horiz div 4);
     place_personnage(nr_pers, 5, ligne, colonne);
     if enregistrer = true then
      begin
       with commande do
        begin
         attitude_pers := 5;
         ligne_pers := ligne;
         colonne_pers := colonne;
        end;
       ecrire_commande_scene(scene_ptr, commande);
      end;
     colonne := colonne + (deplac_horiz div 4);
     place_personnage(nr_pers, 2, ligne, colonne);
     if enregistrer = true then
      begin
       with commande do
        begin
         attitude_pers := 2;
         ligne_pers := ligne;
         colonne_pers := colonne;
        end;
       ecrire_commande_scene(scene_ptr, commande);
      end;
     colonne := colonne + (deplac_horiz div 4);
     place_personnage(nr_pers, 6, ligne, colonne);
     if enregistrer = true then
      begin
       with commande do
        begin
         attitude_pers := 6;
         ligne_pers := ligne;
         colonne_pers := colonne;
        end;
       ecrire_commande_scene(scene_ptr, commande);
      end;
     colonne := colonne_temp + deplac_horiz;
     place_personnage(nr_pers, 2, ligne, colonne);
     if enregistrer = true then
      begin
       with commande do
        begin
         attitude_pers := 2;
         ligne_pers := ligne;
         colonne_pers := colonne;
        end;
       ecrire_commande_scene(scene_ptr, commande);
```

```
      end;
    end
   else
    begin
     colonne_temp := colonne;
     colonne := colonne + (deplac_horiz div 4);
     place_personnage(nr_pers, 7, ligne, colonne);
     if enregistrer = true then
      begin
       with commande do
        begin
         attitude_pers := 7;
         ligne_pers := ligne;
         colonne_pers := colonne;
        end;
       ecrire_commande_scene(scene_ptr, commande);
      end;
     colonne := colonne + (deplac_horiz div 4);
     place_personnage(nr_pers, 4, ligne, colonne);
     if enregistrer = true then
      begin
       with commande do
        begin
         attitude_pers := 4;
         ligne_pers := ligne;
         colonne_pers := colonne;
        end;
       ecrire_commande_scene(scene_ptr, commande);
      end;
     colonne := colonne + (deplac_horiz div 4);
     place_personnage(nr_pers, 8, ligne, colonne);
     if enregistrer = true then
      begin
       with commande do
        begin
         attitude_pers := 8;
         ligne_pers := ligne;
         colonne_pers := colonne;
        end;
       ecrire_commande_scene(scene_ptr, commande);
      end;
     colonne := colonne_temp + deplac_horiz;
     place_personnage(nr_pers, 4, ligne, colonne);
     if enregistrer = true then
      begin
       with commande do
        begin
         attitude_pers := 4;
         ligne_pers := ligne;
         colonne_pers := colonne;
        end;
       ecrire_commande_scene(scene_ptr, commande);
      end;
    end
  else
   begin
    colonne := colonne + deplac_horiz;
    place_personnage(nr_pers, attitude, ligne, colonne);
    if enregistrer = true then
```

```
            begin
             with commande do
              begin
               attitude_pers := attitude;
               ligne_pers := ligne;
               colonne_pers := colonne;
              end;
             ecrire_commande_scene(scene_ptr, commande);
            end;
          end;
         point_1.h := point_2.h;
        end;
       if abs(deplac_vertic) >= contrainte_vertic then
        begin
         ligne := ligne + deplac_vertic;
         place_personnage(nr_pers, attitude, ligne, colonne);
         if enregistrer = true then
          begin
           with commande do
            begin
             attitude_pers := attitude;
             ligne_pers := ligne;
             colonne_pers := colonne;
            end;
           ecrire_commande_scene(scene_ptr, commande);
          end;
         point_1.v := point_2.v;
        end;
      until GetNextEvent(MDownMask, evenement) = true;
      ShowCursor;
      commande.colonne_pers := evenement.where.v;
      commande.ligne_pers := evenement.where.h;
      commande.num_commande := relache_pers;
      if (parametres.type_trace = 3) or (parametres.type_trace = 4) then
       begin
        if trace = nil then
         trace := preparer_fich_trace(parametres);
        trace_commande(trace, evenement.when, -1, commande);
       end;
    end; {suivre_souris}

begin {creer_nouv_scene}
 item_non_accessible(enr_autom);
 item_non_accessible(lireparam);
 item_non_accessible(enregparam);
 stop_creer := false;
 remplir_tableau_lignes;
 ligne_courante := 1;
 pers_courant := 1;
 affiche_ligne_commandes(tab_lignes_commandes[ligne_courante]);
 if (nbre_pers_crees > 0) and (tab_pers[pers_courant].tete <> nil) then
  affiche_image(1, tab_pers[pers_courant].tete);
 enregistrer := parametres.enreg_autom;
 if enregistrer = true then
  begin
   if commandes_enregistrer[ligne_courante] <> 0 then
    voyant(commandes_enregistrer[ligne_courante]);
   a_enregistre := true;
   if scene_ptr = nil then
```

```
      begin
       num_scene := lire_nbre_scenes_enf(parametres.nom_enf) + 1 ;
       ecrire_reference_scene_enf_nr(parametres.nom_enf, num_scene, reference_scene);
       scene_ptr := preparer_scene_enreg(parametres.nom_enf, num_scene, reference_scene);
      end
     else
      effacer_fin_scene_enreg(scene_ptr);
    end
   else
    a_enregistre := false;
  demarrer_musique;
  if (joue_musique = true) and (commandes_musique[ligne_courante] <> 0) then
   voyant(commandes_musique[ligne_courante]);
  while (stop_creer = false) and (quitter_prog = false) do
   begin
    if (parametres.type_trace = 3) or (parametres.type_trace = 4) then
     begin
      if trace = nil then
       trace := preparer_fich_trace(parametres);
      trace_2 := trace;
     end
    else
     trace_2 := nil;
    case parametres.mode_util of
     1 :
      commande := prend_commande_souris(nil, trace_2);
     2 :
      commande := prend_commande_fleches(nil, trace_2);
     3 :
      commande := prend_commande_bal(nil, trace_2);
    end;
    num_com := commande.num_commande;
    case num_com of
     pomme..trace_tout :
      begin
       traiter_choix_menu(num_com);
       if num_com = commandes then
        begin
         remplir_tableau_lignes;
         ligne_courante := 1 ;
         affiche_ligne_commandes(tab_lignes_commandes[ligne_courante]);
         if (nbre_pers_crees > 0) and (tab_pers[pers_courant].tete <> nil) then
          affiche_image(1 , tab_pers[pers_courant].tete);
         if (enregistrer = true) and (commandes_enregistrer[ligne_courante] <> 0) then
          voyant(commandes_enregistrer[ligne_courante]);
         if (joue_musique = true) and (commandes_musique[ligne_courante] <> 0) then
          voyant(commandes_musique[ligne_courante]);
        end;
      end;
     gauche :
      begin
       deplace_gauche(pers_courant);
       if enregistrer = true then
        begin
         commande.num_personnage := pers_courant;
         ecrire_commande_scene(scene_ptr, commande);
        end;
      end;
     droite :
```

```
    begin
     deplace_droite(pers_courant);
     if enregistrer = true then
      begin
       commande.num_personnage := pers_courant;
       ecrire_commande_scene(scene_ptr, commande);
      end;
    end;
  musique:
    begin
     voyant(commandes_musique[ligne_courante]);
     if joue_musique = false then
      demarrer_musique
     else
      terminer_musique;
    end;
  change_pers:
    begin
     if pers_courant = nbre_pers_crees then
      pers_courant := 1
     else
      pers_courant := pers_courant + 1;
     affiche_image(1, tab_pers[pers_courant].tete);
    end;
  change_com:
    begin
     if ligne_courante = nbre_lignes then
      ligne_courante := 1
     else
      ligne_courante := ligne_courante + 1;
     posit_cur := pos_curseur;
     affiche_ligne_commandes(tab_lignes_commandes[ligne_courante]);
     affiche_image(1, tab_pers[pers_courant].tete);
     if (parametres.mode_util = 2) or (parametres.mode_util = 3) then
      place_curseur(posit_cur);
     if (enregistrer = true) and (commandes_enregistrer[ligne_courante] <> 0) then
      voyant(commandes_enregistrer[ligne_courante]);
     if (joue_musique = true) and (commandes_musique[ligne_courante] <> 0) then
      voyant(commandes_musique[ligne_courante]);
    end;
  entree_sortie:
    begin
     if tab_pers[pers_courant].pers_visible = true then
      sortir_scene(pers_courant)
     else
      entrer_scene(pers_courant);
     if enregistrer = true then
      begin
       commande.num_personnage := pers_courant;
       ecrire_commande_scene(scene_ptr, commande);
      end;
    end;
  haut:
    begin
     deplace_haut(pers_courant);
     if enregistrer = true then
      begin
       commande.num_personnage := pers_courant;
       ecrire_commande_scene(scene_ptr, commande);
```

```
      end;
    end;
  bas:
    begin
      deplace_bas(pers_courant);
      if enregistrer = true then
        begin
          commande.num_personnage := pers_courant;
          ecrire_commande_scene(scene_ptr, commande);
        end;
    end;
  debout:
    begin
      lever(pers_courant);
      if enregistrer = true then
        begin
          commande.num_personnage := pers_courant;
          ecrire_commande_scene(scene_ptr, commande);
        end;
    end;
  assis:
    begin
      asseoir(pers_courant);
      if enregistrer = true then
        begin
          commande.num_personnage := pers_courant;
          ecrire_commande_scene(scene_ptr, commande);
        end;
    end;
  couche:
    begin
      coucher(pers_courant);
      if enregistrer = true then
        begin
          commande.num_personnage := pers_courant;
          ecrire_commande_scene(scene_ptr, commande);
        end;
    end;
  pivote_gauche:
    begin
      tourne_gauche(pers_courant);
      if enregistrer = true then
        begin
          commande.num_personnage := pers_courant;
          ecrire_commande_scene(scene_ptr, commande);
        end;
    end;
  pivote_droite:
    begin
      tourne_droite(pers_courant);
      if enregistrer = true then
        begin
          commande.num_personnage := pers_courant;
          ecrire_commande_scene(scene_ptr, commande);
        end;
    end;
  agite:
    begin
      agiter(pers_courant);
```

```
        if enregistrer = true then
         begin
          commande.num_personnage := pers_courant;
          ecrire_commande_scene(scene_ptr, commande);
         end;
       end;
      enregistre:
       begin
        voyant(commandes_enregistrer[ligne_courante]);
        if enregistrer = true then
         enregistrer := false
        else
         begin
          enregistrer := true;
          if a_enregistre = false then
           begin
            a_enregistre := true;
            if scene_ptr = nil then
             begin
              num_scene := lire_nbre_scenes_enf(parametres.nom_enf) + 1;
              ecrire_reference_scene_enf_nr(parametres.nom_enf, num_scene, reference_scene);
              scene_ptr := preparer_scene_enreg(parametres.nom_enf, num_scene, reference_scene);
             end
            else
             effacer_fin_scene_enreg(scene_ptr);
           end;
          enregistrer_positions;
         end;
       end;
      clique_pers:
       begin
        pers_courant := commande.num_personnage;
        affiche_image(1, tab_pers[pers_courant].tete);
        point_clic.v := commande.ligne_pers;
        point_clic.h := commande.colonne_pers;
        suivre_souris(commande.num_personnage, point_clic);
       end;
      stop:
       begin
        num_com := fenetre_ok_annuler(quitter_animation_t, stop_i, parametres.mode_util, trace_2);
        stop_creer := (num_com = ok);
       end;
     end;
   end;
  if joue_musique = true then
   terminer_musique;
  if a_enregistre = true then
   begin
    nouv_nom_scene := fenetre_chaine_car(donne_nom_scene_t, lire_nom_scene(scene_ptr));
    ecrire_nom_scene(scene_ptr, nouv_nom_scene);
   end;
  if scene_ptr <> nil then
   terminer_scene_enreg(scene_ptr);
  for pers_courant := 1 to nbre_pers_crees do
   efface_personnage(pers_courant);
  for ligne_courante := 1 to nbre_lignes do
   detruit_ligne_commandes(tab_lignes_commandes[ligne_courante]);
  item_accessible(enr_autom);
  item_accessible(lireparam);
```

```
  item_accessible(enregparam);
end; {creer_nouv_scene}


procedure Traiter_modification;

 var
  i, j, number : integer;
  fini, passe : boolean;
  quelcom : tcommande;
  tab_comm, tab_ill : ttab_com;
  line_comm : tligne_commandes;
  trace_2 : Tfich_trace_ptr;


begin
 passe := false;
 if pas3comd = false then
  DETRUIT_PERSONNAGE(nbre_pers_crees);
 tab_comm[1] := ok;
 tab_ill[1] := ok_ti;
 for i := 1 to nbre_pers_crees do
  begin
   tab_comm[i + 1] := 1000 + i;
   tab_ill[i + 1] := effacer_ti;
  end;
 for i := nbre_pers_crees + 1 to 4 do
  begin
   tab_comm[i + 1] := 0;
   tab_ill[i + 1] := 0;
  end;
 line_comm := DEFINI_LIGNE_COMMANDES(5, tab_comm, tab_ill, nil);
 AFFICHE_LIGNE_COMMANDES(line_comm);
 fini := false;
 while (fini = false) and (quitter_prog = false) do
  begin
   if (parametres.type_trace = 2) or (parametres.type_trace = 4) then
    begin
     if trace = nil then
      trace := preparer_fich_trace(parametres);
     trace_2 := trace;
    end
   else
    trace_2 := nil;
   case parametres.mode_util of
    1:
     quelcom := PREND_COMMANDE_SOURIS(nil, trace_2);
    2:
     quelcom := PREND_COMMANDE_FLECHES(nil, trace_2);
    3:
     quelcom := PREND_COMMANDE_BAL(nil, trace_2);
    otherwise
   end;
   case quelcom.num_commande of
    pomme..trace_tout:
     TRAITER_CHOIX_MENU(quelcom.num_commande);
    1001, 1002, 1003, 1004:
     begin
      passe := true;
```

```
        number := quelcom.num_commande - 1000;
        DETRUIT_PERSONNAGE(number);
        if (pas3comd = false) and (place_tab <> 1) then
         place_tab := place_tab - 1;
        pas3comd := false;
        for j := 1 to nbre_pers_crees do
         PLACE_PERSONNAGE(j, 1, ligne, cases[j + 1]);
        tab_comm[1] := ok;
        tab_ill[1] := ok_ti;
        for i := 1 to nbre_pers_crees do
         begin
          tab_comm[i + 1] := 1000 + i;
          tab_ill[i + 1] := effacer_ti;
         end;
        for i := nbre_pers_crees + 1 to 4 do
         begin
          tab_comm[i + 1] := 0;
          tab_ill[i + 1] := 0;
         end;
        line_comm := DEFINI_LIGNE_COMMANDES(5, tab_comm, tab_ill, nil);
        AFFICHE_LIGNE_COMMANDES(line_comm);
       end;
      ok:
       fini := true;
      otherwise
     end;
    end;
  DETRUIT_LIGNE_COMMANDES(line_comm);
end;



procedure choisir_personnages;


  var
   ecranfixe: pichandle;
   fin: boolean;
   i, defilpers: integer;
   pers, centre_ecran, line, hauteur_pers: integer;
   comd: tcommande;
   tab_com, tab_illustr: ttab_com;
   ligne_com: tligne_commandes;
   trace_2: Tfich_trace_ptr;


 begin
  for i := 0 to 4 do
   cases[i + 1] := (i * 102) + 51;
{afficher le décor}
  ecranfixe := LIRE_ECRAN_FIXE_NR(3);
  PLACE_DECOR(ecranfixe);
{afficher la ligne de commandes}
  tab_com[1] := precedent;
  tab_com[2] := suivant;
  tab_com[3] := prendre;
  tab_com[4] := modification;
  tab_com[5] := ok;
```

```
    tab_illustr[1] := precedent_ti;
    tab_illustr[2] := suivant_ti;
    tab_illustr[3] := prendre_ti;
    tab_illustr[4] := modifier_ti;
    tab_illustr[5] := ok_ti;
    ligne_com := DEFINI_LIGNE_COMMANDES(5, tab_com, tab_illustr, nil);
    AFFICHE_LIGNE_COMMANDES(ligne_com);
{afficher les personnages}
    for i := 1 to (nbre_pers_crees) do
     begin
      PLACE_PERSONNAGE(i, 1, ligne, cases[i + 1]);
     end;  { i = nbre_pers_crees}
    if nbre_pers_crees <> 4 then
     begin
      pas3comd := false;
      place_tab := nbre_pers_crees + 1;
      if pers_cour > 0 then
       pers_cour := pers_cour - 1;
      defilpers := DEFILER_PERSONNAGES(true, place_tab);
     end
    else
     begin
      pas3comd := true;
      place_tab := 4;
     end;
    centre_ecran := (rectangle_limite.bottom - rectangle_limite.top) div 2;
    with tab_pers[place_tab].tab_attitudes[1].Bounds do
     hauteur_pers := bottom - top;
    line := centre_ecran + (hauteur_pers div 2);
    fin := false;
    while (fin = false) and (quitter_prog = false) do
     begin
      if (parametres.type_trace = 2) or (parametres.type_trace = 4) then
       begin
        if trace = nil then
         trace := preparer_fich_trace(parametres);
        trace_2 := trace;
       end
      else
       trace_2 := nil;
      case parametres.mode_util of
       1:
        comd := PREND_COMMANDE_SOURIS(nil, trace_2);
       2:
        comd := PREND_COMMANDE_FLECHES(nil, trace_2);
       3:
        comd := PREND_COMMANDE_BAL(nil, trace_2);
       otherwise
      end;
      case comd.num_commande of
       pomme..trace_tout:
        TRAITER_CHOIX_MENU(comd.num_commande);
       precedent:
        if pas3comd = false then
         begin
          defilpers := DEFILER_PERSONNAGES(false, place_tab);
         end;
       suivant:
        if pas3comd = false then
```

```
        begin
          defilpers := DEFILER_PERSONNAGES(true, place_tab);
        end;
      prendre:
        begin
          if pas3comd = false then
            begin
              montre_lapin;
              CREE_PERS_COMPLET(place_tab, defilpers);
              cache_lapin;
              PLACE_PERSONNAGE(place_tab, 1, ligne, cases[place_tab + 1]);
              if place_tab <> 4 then
                begin
                  place_tab := place_tab + 1;
                  CREE_PERS_FACE(place_tab, defilpers);
                  PLACE_PERSONNAGE(place_tab, 1, line, cases[1]);
                end
              else
                pas3comd := true;
            end;
        end;
      ok:
        begin
          fin := true;
          for i := 1 to nbre_pers_crees do
            EFFACE_PERSONNAGE(i);
          if pas3comd = false then
            DETRUIT_PERSONNAGE(place_tab);
        end;
      modification:
        begin
          if nbre_pers_crees <> 1 then
            begin
              TRAITER_MODIFICATION;
              AFFICHE_LIGNE_COMMANDES(ligne_com);
              if pas3comd = false then
                begin
                  if pers_cour > 0 then
                    pers_cour := pers_cour - 1;
                  defilpers := DEFILER_PERSONNAGES(true, place_tab);
                end;
            end;
        end;
      otherwise
    end;
  end;
  DETRUIT_LIGNE_COMMANDES(ligne_com);
end;



function choisir_decor;


  var
    defildecor : integer;
    fin : boolean;
```

```
    comd: tcommande;
    tableau_com, tableau_illustr: ttab_com;
    ligne_comde: tligne_commandes;
    trace_2: Tfich_trace_ptr;


 begin
{afficher le premier décor }
   if decor_cour >= 0 then
    decor_cour := decor_cour - 1;
   defildecor := DEFILER_DECORS(true);
{afficher la ligne de commandes}
   tableau_com[1] := precedent;
   tableau_com[2] := suivant;
   tableau_com[3] := ok;
   tableau_illustr[1] := precedent_ti;
   tableau_illustr[2] := suivant_ti;
   tableau_illustr[3] := ok_ti;
   ligne_comde := DEFINI_LIGNE_COMMANDES(3, tableau_com, tableau_illustr, nil);
   AFFICHE_LIGNE_COMMANDES(ligne_comde);
   fin := false;
   while (fin = false) and (quitter_prog = false) do
    begin
     if (parametres.type_trace = 2) or (parametres.type_trace = 4) then
      begin
       if trace = nil then
        trace := preparer_fich_trace(parametres);
       trace_2 := trace;
      end
     else
      trace_2 := nil;
     case parametres.mode_util of
      1:
       comd := PREND_COMMANDE_SOURIS(nil, trace_2);
      2:
       comd := PREND_COMMANDE_FLECHES(nil, trace_2);
      3:
       comd := PREND_COMMANDE_BAL(nil, trace_2);
      otherwise
     end;
     case comd.num_commande of
      pomme..trace_tout:
       TRAITER_CHOIX_MENU(comd.num_commande);
      precedent:
       defildecor := DEFILER_DECORS(false);
      suivant:
       defildecor := DEFILER_DECORS(true);
      ok:
       begin
        choisir_decor := defildecor;
        fin := true;
       end;
     end;
    end;
  DETRUIT_LIGNE_COMMANDES(ligne_comde);
 end;


 function choisir_musique;
```

```
  var
   defilmusique : integer;
   fin : boolean;
   comd : tcommande;
   tableau_com, tableau_illustr : ttab_com;
   ligne_comde : tligne_commandes;
   trace_2 : Tfich_trace_ptr;


 begin
{afficher le premier écran_musique }
  if musique_cour > 0 then
   musique_cour := musique_cour - 1;
  defilmusique := DEFILER_ECRANS_MUSIQUE(true);
  jeter_musique;
  charger_musique(lire_musique_nr(defilmusique));
{afficher la ligne de commandes}
  tableau_com[1] := precedent;
  tableau_com[2] := suivant;
  tableau_com[3] := ecouter;
  tableau_com[4] := ok;
  tableau_illustr[1] := precedent_ti;
  tableau_illustr[2] := suivant_ti;
  tableau_illustr[3] := ecouter_ti;
  tableau_illustr[4] := ok_ti;
  ligne_comde := DEFINI_LIGNE_COMMANDES(4, tableau_com, tableau_illustr, nil);
  AFFICHE_LIGNE_COMMANDES(ligne_comde);
  fin := false;
  while (fin = false) and (quitter_prog = false) do
   begin
    if (parametres.type_trace = 2) or (parametres.type_trace = 4) then
     begin
      if trace = nil then
       trace := preparer_fich_trace(parametres);
      trace_2 := trace;
     end
    else
     trace_2 := nil;
    case parametres.mode_util of
     1:
      comd := PREND_COMMANDE_SOURIS(nil, trace_2);
     2:
      comd := PREND_COMMANDE_FLECHES(nil, trace_2);
     3:
      comd := PREND_COMMANDE_BAL(nil, trace_2);
     otherwise
    end;
    case comd.num_commande of
     pomme..trace_tout:
      TRAITER_CHOIX_MENU(comd.num_commande);
     precedent:
      begin
       defilmusique := DEFILER_ECRANS_MUSIQUE(false);
       if joue_musique = true then
        voyant(3);
       jeter_musique;
       charger_musique(lire_musique_nr(defilmusique));
      end;
```

```
      suivant:
       begin
        defilmusique := DEFILER_ECRANS_MUSIQUE(TRUE);
        if joue_musique = true then
         voyant(3);
        jeter_musique;
        charger_musique(lire_musique_nr(defilmusique));
       end;
      ecouter:
       begin
        voyant(3);
        if joue_musique = false then
         demarrer_musique
        else
         terminer_musique;
       end;
      ok:
       begin
        if joue_musique = true then
         terminer_musique;
        choisir_musique := defilmusique;
        fin := true;
       end;
     end;
   end;
  DETRUIT_LIGNE_COMMANDES(ligne_comde);
end; {choisir_musique}


function Choisir_scene;

 var
  tableau_comm, tableau_illus: ttab_com;
  ligne_comm: tligne_commandes;
  quelscene: tscene_ref;
  defilscene, reponse, i, combien: integer;
  fin: boolean;
  comd: tcommande;
  trace_2: Tfich_trace_ptr;


begin
 fin := false;
 if scene_cour > 0 then
  scene_cour := scene_cour - 1;
 defilscene := DEFILER_SCENES(true, parametres.nom_enf, quelscene);
 tableau_comm[1] := precedent;
 tableau_comm[2] := suivant;
 tableau_comm[3] := ok;
 tableau_comm[4] := jeter;
 tableau_comm[5] := stop;
 tableau_illus[1] := precedent_ti;
 tableau_illus[2] := suivant_ti;
 tableau_illus[3] := ok_ti;
 tableau_illus[4] := jeter_ti;
 tableau_illus[5] := stop_i;
 ligne_comm := DEFINI_LIGNE_COMMANDES(5, tableau_comm, tableau_illus, nil);
 AFFICHE_LIGNE_COMMANDES(ligne_comm);
 while (fin = false) and (quitter_prog = false) do
```

```
begin
 if (parametres.type_trace = 3) or (parametres.type_trace = 4) then
  begin
   if trace = nil then
    trace := preparer_fich_trace(parametres);
   trace_2 := trace;
  end
 else
  trace_2 := nil;
 case parametres.mode_util of
  1:
   comd := PREND_COMMANDE_SOURIS(nil, trace_2);
  2:
   comd := PREND_COMMANDE_FLECHES(nil, trace_2);
  3:
   comd := PREND_COMMANDE_BAL(nil, trace_2);
  otherwise
 end;
 case comd.num_commande of
  pomme..trace_tout:
   begin
    TRAITER_CHOIX_MENU(comd.num_commande);
    if (comd.num_commande = enregparam) or (comd.num_commande = lireparam) then
     begin
      fin := true;
      Choisir_scene := nil;
     end;
   end;
  precedent:
   defilscene := DEFILER_SCENES(false, parametres.nom_enf, quelscene);
  suivant:
   defilscene := DEFILER_SCENES(true, parametres.nom_enf, quelscene);
  ok:
   begin
    fin := true;
    Choisir_scene := PREPARER_SCENE_ENREG(parametres.nom_enf, defilscene, quelscene);
    montre_lapin;
    if quelscene.num_pers_1 <> 0 then
     CREE_PERS_COMPLET(1, quelscene.num_pers_1);
    if quelscene.num_pers_2 <> 0 then
     CREE_PERS_COMPLET(2, quelscene.num_pers_2);
    if quelscene.num_pers_3 <> 0 then
     CREE_PERS_COMPLET(3, quelscene.num_pers_3);
    if quelscene.num_pers_4 <> 0 then
     CREE_PERS_COMPLET(4, quelscene.num_pers_4);
    change_lapin;
    jeter_musique;
    charger_musique(lire_musique_nr(quelscene.num_musique));
    cache_lapin;
    RETRACE_ECRAN;
   end;
  stop:
   begin
    fin := true;
    Choisir_scene := nil;
   end;
  jeter:
   begin
    reponse := FENETRE_OK_ANNULER(jeter_t, jeter_i, parametres.mode_util, trace_2);
```

```
        retrace_ecran;
        if reponse = ok then
         begin
          montre_lapin;
          EFFACER_SCENE_ENF_NR(parametres.nom_enf, defilscene);
          change_lapin;
          combien := LIRE_NBRE_SCENES_ENF(parametres.nom_enf);
          cache_lapin;
          if combien = 0 then
           begin
            choisir_scene := nil;
            fin := true;
           end
          else
           defilscene := DEFILER_SCENES(true, parametres.nom_enf, quelscene);
         end;
       end;
      otherwise
     end;
    end;
  DETRUIT_LIGNE_COMMANDES(ligne_comm);
  for i := 1 to nbre_pers_crees do
   EFFACE_PERSONNAGE(i);
end;



function Visualiser_scene;

 var
  tab_comd, tab_ill: ttab_com;
  ligne_comd: tligne_commandes;
  fini, pause_ok: boolean;
  commande: tcommande;
  sauve_ptr: Tscene_enreg_ptr;
  reponse: integer;
  trace_2: Tfich_trace_ptr;

begin
 ITEM_NON_ACCESSIBLE(enregparam);
 ITEM_NON_ACCESSIBLE(lireparam);
 fini := false;
 tab_comd[1] := recreer;
 tab_comd[2] := musique;
 tab_comd[3] := pause;
 tab_comd[4] := stop;
 tab_ill[1] := recreer_ti;
 tab_ill[2] := musique_ti;
 tab_ill[3] := pause_ti;
 tab_ill[4] := stop_i;
 ligne_comd := DEFINI_LIGNE_COMMANDES(4, tab_comd, tab_ill, nil);
 AFFICHE_LIGNE_COMMANDES(ligne_comd);
 pause_ok := false;
 demarrer_musique;
 if joue_musique = true then
  voyant(2);
```

```
while (fini = false) and (quitter_prog = false) do
 begin
  if (parametres.type_trace = 3) or (parametres.type_trace = 4) then
   begin
    if trace = nil then
     trace := preparer_fich_trace(parametres);
    trace_2 := trace;
   end
  else
   trace_2 := nil;
  case parametres.mode_util of
   1:
    commande := PREND_COMMANDE_SOURIS(scene_ptr, trace_2);
   2:
    commande := PREND_COMMANDE_FLECHES(scene_ptr, trace_2);
   3:
    commande := PREND_COMMANDE_BAL(scene_ptr, trace_2);
   otherwise
  end;
  case commande.num_commande of
   pomme..trace_tout:
    TRAITER_CHOIX_MENU(commande.num_commande);
   recreer:
    begin
     reponse := FENETRE_OK_ANNULER(recreer_t, recreer_i, parametres.mode_util, trace_2);
     if reponse = ok then
      begin
       fini := true;
       if pause_ok = true then
        scene_ptr := sauve_ptr;
       visualiser_scene := recreer;
      end;
    end;
   musique:
    begin
     voyant(2);
     if joue_musique = false then
      demarrer_musique
     else
      terminer_musique;
    end;
   pause:
    begin
     voyant(3);
     if pause_ok = false then
      begin
       pause_ok := true;
       sauve_ptr := scene_ptr;
       scene_ptr := nil;
      end
     else
      begin
       pause_ok := false;
       scene_ptr := sauve_ptr;
      end;
    end;
   stop:
    begin
     reponse := FENETRE_OK_ANNULER(arreter_regarder_t, stop_i, parametres.mode_util, trace_2);
```

```
          if reponse = ok then
           begin
            fini := true;
            if pause_ok = true then
             scene_ptr := sauve_ptr;
            TERMINER_SCENE_ENREG(scene_ptr);
            visualiser_scene := stop;
           end;
         end;
       place_pers:
         with commande do
          place_personnage(num_personnage, attitude_pers, ligne_pers, colonne_pers);
       efface_pers:
         efface_personnage(commande.num_personnage);
       gauche:
         deplace_gauche(commande.num_personnage);
       droite:
         deplace_droite(commande.num_personnage);
       entree_sortie:
         begin
          if tab_pers[commande.num_personnage].pers_visible = true then
           sortir_scene(commande.num_personnage)
          else
           entrer_scene(commande.num_personnage);
         end;
       haut:
         deplace_haut(commande.num_personnage);
       bas:
         deplace_bas(commande.num_personnage);
       debout:
         lever(commande.num_personnage);
       assis:
         asseoir(commande.num_personnage);
       couche:
         coucher(commande.num_personnage);
       pivote_gauche:
         tourne_gauche(commande.num_personnage);
       pivote_droite:
         tourne_droite(commande.num_personnage);
       agite:
         agiter(commande.num_personnage);
       otherwise
      end;
     end;
    if joue_musique = true then
     terminer_musique;
    DETRUIT_LIGNE_COMMANDES(ligne_comd);
    ITEM_ACCESSIBLE(enregparam);
    ITEM_ACCESSIBLE(lireparam);
  end;

end. {unit creativite}
```

```
program OrdiTheatre;

 uses
  fichiers, enregistrement, musique, animation, gestion_menus, ligne_de_commandes, commandes,
       gestion_dialogue, ecrans, creativite;

 var
  ligne_com : Tligne_commandes;
  tab_com, tab_illustr : Ttab_com;
  ecran : PicHandle;
  commande : Tcommande;
  num_com : integer;
  pointeur_scene : Tscene_enreg_ptr;
  reference_nulle : Tscene_ref;
  main_curs : CursHandle;
  trace_2 : Tfich_trace_ptr;

 procedure initialiser;

 begin
  FlushEvents(EveryEvent, 0);
  main_curs := GetCursor(1000);
  HNoPurge(Handle(main_curs));
  SetCursor(main_curs^^);
  ouvrir_fichiers;
  init_musique;
  init_ligne_commandes;
  init_animation;
  init_menus;
  init_commandes;
  init_ecrans;
  fenetre_a_propos_de(240);
  tab_com[1] := regarder_scene;
  tab_com[2] := creer_scene;
  tab_illustr[1] := regarder_scene_t;
  tab_illustr[2] := creer_scene_t;
  ligne_com := defini_ligne_commandes(2, tab_com, tab_illustr, nil);
  parametres.nom_enf := fenetre_chaine_car(demande_nom_t, '');
  parametres := lire_parametres_enfant(parametres.nom_enf);
  if parametres.enreg_autom = true then
   mettre_marque(enr_autom);
  vitesse_defil := parametres.vitesse_def;
  case parametres.mode_util of
   1 :
    begin
     mettre_marque(mode_souris);
     cache_souris := false;
    end;
   2 :
    begin
     mettre_marque(mode_fleches);
     cache_souris := true;
     HideCursor;
    end;
   3 :
    begin
     mettre_marque(mode_bal);
     cache_souris := true;
     HideCursor;
```

```
    end;
  end;
  case parametres.type_trace of
   1:
    begin
     mettre_marque(pas_trace);
     trace := nil;
    end;
   2:
    begin
     mettre_marque(trace_sel);
     trace := preparer_fich_trace(parametres);
    end;
   3:
    begin
     mettre_marque(trace_anim);
     trace := nil;
    end;
   4:
    begin
     mettre_marque(trace_tout);
     trace := preparer_fich_trace(parametres);
    end;
  end;
  ecran := lire_ecran_fixe_nr(1);
  place_decor(ecran);
  affiche_ligne_commandes(ligne_com);
  quitter_prog := false;
end; {initialiser}

function traiter_regarder_scene (var pointeur_scene : Tscene_enreg_ptr): integer;

 var
  num_com : integer;
  i: integer;

begin
 repeat
  pointeur_scene := choisir_scene;
  if (pointeur_scene <> nil) and (quitter_prog = false) then
   num_com := visualiser_scene(pointeur_scene)
  else
   num_com := stop;
 until (pointeur_scene = nil) or (num_com = recreer) or (quitter_prog = true);
 if num_com = stop then
  begin
   while nbre_pers_crees <> 0 do
    detruit_personnage(1);
   jeter_musique;
  end;
 traiter_regarder_scene := num_com;
end; {traiter_regarder_scene}

procedure traiter_creer_scene (pointeur_scene : Tscene_enreg_ptr);

 var
  ligne_com : Tligne_commandes;
  tab_com, tab_illustr : Ttab_com;
  commande : Tcommande;
```

```
    ecran, decor_pic : PicHandle;
    num_decor_choisi : integer;
    num_musique_choisie : integer;
    tab_pers_choisis : array[1..4] of integer;
    num_scene : integer;
    reference_scene : Tscene_ref;
    stop_creer : boolean;
    pas_decor, pas_pers, pas_musique, continuer : boolean;
    i : integer;
    trace_2 : Tfich_trace_ptr;

begin
  tab_com[1] := decors;
  tab_com[2] := personnages;
  tab_com[3] := musique;
  tab_com[4] := creer;
  tab_com[5] := stop;
  tab_illustr[1] := decors_t;
  tab_illustr[2] := personnages_t;
  tab_illustr[3] := musique_t;
  tab_illustr[4] := creer_t;
  tab_illustr[5] := stop_t;
  ligne_com := defini_ligne_commandes(5, tab_com, tab_illustr, nil);
  ecran := lire_ecran_fixe_nr(2);
  place_decor(ecran);
  affiche_ligne_commandes(ligne_com);
  stop_creer := false;
  if pointeur_scene = nil then
    begin
      num_decor_choisi := -1;
      num_musique_choisie := -1
    end
  else
    begin
      num_decor_choisi := pointeur_scene^.reference.num_decor;
      num_musique_choisie := pointeur_scene^.reference.num_musique;
    end;
  repeat
    if (parametres.type_trace = 2) or (parametres.type_trace = 4) then
      begin
        if trace = nil then
          trace := preparer_fich_trace(parametres);
        trace_2 := trace;
      end
    else
      trace_2 := nil;
    case parametres.mode_util of
      1 :
        commande := prend_commande_souris(nil, trace_2);
      2 :
        commande := prend_commande_fleches(nil, trace_2);
      3 :
        commande := prend_commande_bal(nil, trace_2);
    end;
    case commande.num_commande of
      pomme..trace_tout :
        traiter_choix_menu(commande.num_commande);
      decors :
        num_decor_choisi := choisir_decor;
```

```
    personnages:
     choisir_personnages;
    musique:
     num_musique_choisie := choisir_musique;
    creer:
     begin
      if num_decor_choisi = -1 then
       pas_decor := true
      else
       pas_decor := false;
      if num_musique_choisie = -1 then
       pas_musique := true
      else
       pas_musique := false;
      if nbre_pers_crees = 0 then
       pas_pers := true
      else
       pas_pers := false;
      if (pas_decor = true) or (pas_pers = true) or (pas_musique = true) then
       continuer := (fenetre_pas_selectionne(pas_pers, pas_decor, pas_musique, parametres.mode_util,
     trace_2) = ok)
      else
       continuer := true;
      if continuer = true then
       begin
        for i := 1 to nbre_pers_crees do
         tab_pers_choisis[i] := tab_pers[i].num_pers_disque;
        for i := nbre_pers_crees + 1 to max_personnages do
         tab_pers_choisis[i] := 0;
        with reference_scene do
         begin
          nom_scene := '';
          num_pers_1 := tab_pers_choisis[1];
          num_pers_2 := tab_pers_choisis[2];
          num_pers_3 := tab_pers_choisis[3];
          num_pers_4 := tab_pers_choisis[4];
          num_decor := num_decor_choisi;
          num_musique := num_musique_choisie;
         end;
        decor_pic := lire_decor_nr(num_decor_choisi);
        place_decor(decor_pic);
        creer_nouv_scene(nil, reference_scene);
       end;
     end;
    stop:
     stop_creer := (fenetre_ok_annuler(demande_stop_t, stop_i, parametres.mode_util, trace_2) = OK);
   end;
   if (commande.num_commande < pomme) and (commande.num_commande <> stop) and (quitter_prog = false)
      then
    begin
     ecran := lire_ecran_fixe_nr(2);
     place_decor(ecran);
     affiche_ligne_commandes(ligne_com);
    end;
 until (quitter_prog = true) or (stop_creer = true);
 while nbre_pers_crees <> 0 do
  detruit_personnage(1);
 jeter_musique;
 detruit_ligne_commandes(ligne_com);
```

```
  end; {traiter_creer_scene}

begin {OrdiTheatre}
  initialiser;
  repeat
    if (parametres.type_trace = 2) or (parametres.type_trace = 4) then
      begin
        if trace = nil then
          trace := preparer_fich_trace(parametres);
        trace_2 := trace;
      end
    else
      trace_2 := nil;
    case parametres.mode_util of
      1:
        commande := prend_commande_souris(nil, trace_2);
      2:
        commande := prend_commande_fleches(nil, trace_2);
      3:
        commande := prend_commande_bal(nil, trace_2);
    end;
    case commande.num_commande of
      pomme..trace_tout:
        traiter_choix_menu(commande.num_commande);
      regarder_scene:
        if lire_nbre_scenes_enf(parametres.nom_enf) > 0 then
          begin
            num_com := traiter_regarder_scene(pointeur_scene);
            if (num_com = recreer) and (quitter_prog = false) then
              begin
                creer_nouv_scene(pointeur_scene, reference_nulle);
                if quitter_prog = false then
                  traiter_creer_scene(pointeur_scene);
              end;
          end;
      creer_scene:
        traiter_creer_scene(nil);
    end;
    if (commande.num_commande < pomme) and (quitter_prog = false) then
      begin
        ecran := lire_ecran_fixe_nr(1);
        place_decor(ecran);
        affiche_ligne_commandes(ligne_com);
      end;
  until quitter_prog = true;
  if trace <> nil then
    terminer_fich_trace(trace);
  fermer_fichiers;
  CloseWindow(fenetre_lapin);
  CloseWindow(ecran_scene);
  CloseWindow(port_graph);
  InitCursor;
end. {OrdiTheatre}
```

Réalisation, impression et reliures : SPRL Centre de Reprographie - Tél. : 081/22.87.72